PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

MAESTRÍA EN REDES DE COMUNICACIÓN

PERFIL DEL TRABAJO PREVIO LA OBTENCIÓN DEL TÍTULO DE:

MASTER EN REDES DE COMUNICACIÓN

TEMA:

"ESTUDIO DE BALANCEO DE CARGA DE UN SISTEMA DE SOFTWARE LIBRE PARA STREAMING DE ALTA DISPONIBILIDAD: CLUSTER DE JBOSS CON RED5 CON BALANCEO DE CARGA EN AMAZON WEB SERVICES"

AUTOR: JUAN DIEGO CALLE BORJA

DIRECTOR: MSC. FRANCISCO RODRÍGUEZ CLAVIJO

Quito, 2016

Tabla de contenido

Índice de tablas

Índice de ilustraciones

Introducción

Esta tesis recoge los resultados del estudio de balanceo de carga de un sistema de software libre para *streaming* de alta disponibilidad en la nube utilizando *Amazon Web Service.* Para ello se parte de una investigación basada en 4 conceptos fundamentales que generan el resultado final: streaming, *clusters,* software libre y Servicios en la Nube con Amazon Web Services.

Los sistemas de streaming son sistemas que permiten la transmisión y recepción de datos por medio de una computadora en una red. Generalmente, pero no exclusivamente, se lo usa para la transmisión de audio y video. La característica principal del streaming es que posibilita un flujo continuo de datos, y para los casos de Video o Audio significa que admiten adelantar o retroceder los datos recibidos. Por esto es ideal para crear sistemas en tiempo real donde puedan correr programas de video conferencias, chats hasta juegos de uso masivo o MMOGs por sus siglas en inglés (*Massive Multiplayer Online Games*). Esto faculta que varios usuarios se conecten e interaccionen al mismo tiempo ("Streaming", 2016).

Los clusters son arreglos de servidores con una configuración similar, se busca que tenga un *Hardware* o en el caso de ser máquinas virtuales una configuración Homogénea, que nos permiten distribuir la carga y soportar fallos, son usados para alta disponibilidad, alto rendimiento, balanceo de carga y escalabilidad. Esto significa que puede ser usado para permitir un mejor servicio, poder distribuir la carga y en el caso de que se necesite poder crecer ("Cluster (informática)", 2016).

JBoss es el primer servidor de aplicaciones enteramente hecho en software abierto/libre. Tiene una versión comunitaria JBoss AS y una versión de pago JBoss Enterprise Application Plataform. Desde la versión 7 JBoss AS se cambió el nombre a WildFly para evitar confusión entre las versiones comunitarias y las de pago. Es mantenido por la Empresa Red Hat. JBoss fue diseñado para ser utilizado en clusters. El servidor web o contenedor que utiliza JBoss es Apache Tomcat, lo cual hace posible que Red5 corra bajo ese servidor("WildFly", 2016).

Cloud Services o servicios en la nube, son los servicios que permiten mantener la información no en un equipo físico si no en máquinas virtuales o "alquilando" espacio a un proveedor. Lo cual es perfecto para crear una red homogénea de servidores y en el caso de necesitarse, escalar. Los Servicios en la Nube suelen cobrar por uso, datos almacenados, ancho de banda, transmisión de datos. Esto puede ayudar a las empresas pequeñas a solo pagar por lo que están usando y en el caso que necesiten crecer tienen el espacio suficiente ("Computación en la nube", 2016).

AMAZON WEB SERVICES (AWS) son los servicios en la nube más utilizados con el respaldo de la empresa Amazon, por lo que tiene una gama de servicios muy amplia, ideal para proyectos de streaming, sus precios son económicos por lo que es ideal para pequeñas y medianas empresas ("About AWS", 2016).

Con el estudio de estos conceptos y herramientas pude conocer sobre los clusters de alta disponibilidad, la estructura de Red5, y descubrí la necesidad de modificar el código fuente de este último para poder brindar una solución a mi propuesta de trabajar desde el software libre. Para el estudio de Red5 descubrí que un cluster RTM no funciona igual que un cluster dirigido a servicios web. Para ello se requirió la creación de una aplicación Red5 que pueda trasmitir las sesiones RTMP[1] hacia los otros nodos. La documentación de Red5 es escasa y en algunos casos inexistente, dependí de mi habilidad de leer e interpretar su código para entender cómo funcionaba. Me apoyé en la comunidad de Red5 y en los pocos ejemplos de aplicación existentes en la web.

Con este estudio se cumplió el objetivo general de estudiar el balanceo de carga de un sistema de software libre para streaming de alta disponibilidad utilizando cluster de Jboss con Red5 en la nube con Amazon Web Services (AWS). Para ello fue necesario cumplir los objetivos específicos:

- Identificar el estado de arte de las tecnologías streaming de alta disponibilidad.
- Comparar las soluciones de streaming comerciales y abiertas.
- Identificar costos del proveedor de servicios de streaming para vídeo, audio o para soporte de aplicaciones en tiempo real como son los juegos MMO.
- Estudiar el balanceo de carga en la nube.

Implementar aplicaciones de software libre para streaming de alta disponibilidad.

Consecuentemente se comprobó la hipótesis: El balanceo de carga con aplicaciones de software libre en la nube es una solución económicamente viable para streaming en pequeños y medianos proyectos.

Este documento final recoge el proceso, las pruebas, el análisis y las comprobaciones que se exponen en los 5 capítulos y las conclusiones:

Capítulo I. Comprende una introducción hacia los servicios de streaming disponibles en el mercado y los servicios en la nube y herramientas que se utilizaron en esta tesis. En este capítulo se cumplen los objetivos específicos: “Identificar el estado de arte de las tecnologías streaming de alta disponibilidad”, “comparar las soluciones de streaming comerciales y abiertas”, e “identificar costos del proveedor de servicios de streaming para vídeo, audio o para soporte de aplicaciones en tiempo real como son los juegos MMO”.

Capítulo 2. Trata los temas relacionados con la alta disponibilidad: tipos de arquitectura, el manejo de carga y herramientas que operan sobre los clusters. Con ello se logra el objetivo específico: “Estudiar el balanceo de carga en la nube”.

1 RTMP: Es un protocolo de alto rendimiento diseñado para la transmisión con audio, video y datos. Es una tecnología con especificaciones abierta.

Capítulo 3. A partir de este capítulo se expone el proyecto. Va dirigido a describir el proceso de instalación en un cluster. Por tanto se cumple el objetivo de “Implementar aplicaciones de software libre para streaming de alta disponibilidad”.

Capítulo 4. En éste se explica el proceso de empaquetado del Red5 para poder utilizarlo en base al proyecto. Al igual que el anterior capítulo va encaminado a “Implementar aplicaciones de software libre para streaming de alta disponibilidad”.

Capítulo 5. Se muestra la aplicación creada y se hace un análisis comparativo de ésta con respecto a las herramientas del mercado. Se culmina con el objetivo: “Implementar aplicaciones de software libre para streaming de alta disponibilidad” y además se realiza una comparación con las soluciones de streaming comerciales y abiertas.

Por último las conclusiones expresan los descubrimientos, problemas y soluciones que se encontraron a lo largo del proceso de investigación.

Capítulo 1: Avances del streaming en el mercado

En este capítulo se expone la investigación realizada sobre las distintas soluciones para el streaming que existen en el mercado. Por tanto cumple los objetivos específicos: “Identificar el estado de arte de las tecnologías streaming de alta disponibilidad”, “comparar las soluciones de streaming comerciales y abiertas”, e “identificar costos del proveedor de servicios de streaming para vídeo, audio o para soporte de aplicaciones en tiempo real como son los juegos MMO”.

Para ello fueron decisivas la experiencia personal y la búsqueda que permite el internet. Además es necesario detenerse en los avances de las herramientas utilizadas en esta tesis: Red5, JBOSS/Wildfly, Amazon Web Services.

1.1 Servidores de aplicación de streaming

Existen varios tipos de servicio de streaming para publicación de contenido que pueden variar de acuerdo de las necesidades de los usuarios, precios y requerimientos de los sistemas. Se puede acceder a servicios gratuitos con ciertas restricciones de uso, servicios pagados con cantidades distintas de ancho de banda, procesamiento y otras configuraciones, hasta servidores virtuales o servidores dedicados con servicios instalados como los de Adobe Media Server, WowZa o Red5.

En las soluciones comerciales existen Adobe Media Server, que tiene cinco tipos de licencias: *starter* que es introductoria y bastante limitada; la estándar con costos medios, que tiene limitaciones; luego la profesional (Adobe Media Server 5 Professional); la extendida que contiene lo de la profesional, pero para una carga mayor, ambas de precios altos; y un tipo de licencia para servidores en Amazon, que tiene un precio variante de acuerdo a las capacidades del servidor. También hay WOWZA con una licencia que se otorga en base a las instancias que maneja en las primeras 4 instancias. Después de la cuarta instancia bajan los precios de acuerdo a la cantidad de servidores que corran con WOWZA (“Buying guide : Pricing : | Adobe Media Server family”, s/f). Wonza también maneja un tipo de licencia para Amazon Web Service.

Red5 es gratuito y de código abierto y libre. Permite su modificación, tiene una comunidad activa y hay empresas que trabajan directamente y desarrollan con Red5. Hay otras opciones gratuitas y de código libre y abierto por ejemplo con NodeJS, pero sus módulos están en un desarrollo temprano y no están a la par con las otras soluciones. Node.js stream está en la versión 0.12,5. JBoss está diseñado para utilizar balanceo de carga y clusters.

El balanceo de carga permite escalar aplicaciones soportando el crecimiento de usuarios concurrentes. De la misma forma protege las aplicaciones contra ataques distribuidos de denegación de servicio, mejor conocidos como DDoS por sus siglas en inglés (Distributed Denial of Service).

Por otro lado los servicios de streaming de video y audio ocupan gran parte del tráfico de internet. Por ejemplo: en Netflix el servicio de streaming de películas ocupa el 35% del tráfico web a nivel mundial (López Tazón, 2015).

No todos los proveedores son intercambiables ni ofrecen exactamente los mismos servicios. Algunos están orientados más hacia video y/o audio, mientras que otros hacia aplicaciones.

Servicios Web de streaming por suscripción para publicación de contenido

La publicación de contenido en la web se lo puede hacer con mucha facilidad, ya que se tiene una gran cantidad de proveedores, destinados a distintos objetivos y audiencias.

Muchos de proveedores tienen servicios gratuitos, algunos tienen sus costos y sus planes con distintos beneficios y soporte. Los más conocidos son:

YouTube

Con más de mil millones de usuarios se puede decir que es el servicio de publicación de video más popular ("Statistics - YouTube", 2016). Permite que cualquier usuario registrado cree un canal y suba el contenido. Presenta opciones para la creación de canales comerciales en los cuales se puede "monetizar" o recibir remuneraciones por medio de publicidad ("Insertar listas y anuncios - Ayuda de YouTube", s/f).

Al usar la opción de "monetización" uno no puede elegir la publicidad que sale, esto lo elige YouTube bajo sus políticas. Lo que puede implicar un problema para la persona que sube un producto porque está ajena al contenido de la propaganda que se le impone.

Este medio soporta streaming en vivo a través de la creación de eventos con fecha y hora determinados. Se puede manejar distintas configuraciones de privacidad para manejar quien puede ver o no los videos ("Introducción a los eventos en directo - Ayuda de YouTube", s/f).

En definitiva lo único que se necesita es una cuenta verificada y unas aplicaciones extra para poder hacer streaming. Hay varias aplicaciones soportadas para Windows, Mac y hasta Android, lamentablemente ninguna es soportada para Linux ("Set up your live streaming encoder - YouTube Help", s/f), lo que marca una restricción para el software libre.

Vimeo

Es una página web que provee servicio streaming de video orientados a la alta calidad. No son videos con objetivos comerciales, pero si tienen opciones de tipos de cuentas valoradas y gratuitas. Generalmente se orienta a videos de alta calidad y de producción artística ("Vimeo", 2016).

Vimeo no soporta streaming en vivo, tiene restricciones para sus usuarios en relación a la cantidad de archivos que se puede subir en un lapso de tiempo. Esto se vuelve una restricción para los productores de video de bajo presupuesto.

Twitch

Esta es una empresa adquirida por Amazon.com que está enfocada al streaming en vivo, sobre todo dirigida a los eSports ("Twitch - Wikipedia, la enciclopedia libre", 2016). eSports son "deportes" electrónicos, que comprenden torneos y campeonatos de videojuegos. Está orientada a la comunidad de Gamers o jugadores de videojuegos como es conocido en español. En mi experiencia he visto como muchos jugadores *amateurs* utilizan esta plataforma para grabar y compartir sus mejores y hasta en algunos casos peores juegos para aprender, o demostrar sus habilidades, o simplemente por broma.

Similar a YouTube permite generar ganancias por medio de publicidad que tampoco es controlada por los autores de los contenidos subidos.

Ustream

Consiste en una página enfocada a streaming de video profesional. Tiene sus propias herramientas de edición de video. No tiene planes gratuitos, aunque sí permite probar el sistema de forma gratuita por tiempo limitado ("Video Streaming Platforms, Options and Pricing", 2016). Su objetivo son los usuarios corporativos, empresas que desean brindar un streaming serio.

Snapchat

Esta es más una aplicación pues permite subir imágenes y videos para compartir que un servicio web de streaming directamente. Tiene opciones similares a la de YouTube para "monetización". Aunque la idea es generar videos cortos y publicarlos, más como un chat en vivo. Los videos o imágenes se pueden generar con un tiempo de vida limitada y generalmente van acompañados con palabras o frases explicativas sobre esto. Es más un medio social que una imagen corporativa. Aunque algunas empresas usan snapchat para llegar a los usuarios más jóvenes ("Snapchat", 2016).

Periscope

Esta es una herramienta para trasmisión de streaming de propiedad de *Twitter* y por tanto se maneja solo desde celular y una cuenta personal de este servicio web de *microblogging.* Es un servicio gratuito. Permite restringir que usuarios pueden ver los vídeos.(Inc, s/f)

	Plan Gratuito	Planes de Pago Para Publicación	Publicación de Videos	Streaming en Vivo	Generación de Ingresos por Publicidad
YouTube	Sí	No	Sí	Sí	Sí
Vimeo	Sí	Sí. Las cuentas gratuitas tienen límite de videos por semanas para subir. Al pagar los diversos planes permite subir mayor cantidad de videos con mayor tamaño.	Sí	No	No
Twitch	Sí	Tiene una opción llamada Twitch Turbo que permite a los usuarios publicar sus videos sin publicidad. Amplía la capacidad de almacenamiento. ("Twitch Payment Information", 2016)	No	Sí	Sí
Ustream	No. Tiene un "Trial" o plan de prueba por 1 mes	Sí	Sí	Sí	No
Snapchat	Sí	No	Sí	Sí	Sí
Periscope	Si	No	No	Si	No

Tabla 1: Cuadro comparativo de servicios web de streaming de Video. Elab. Propia.

1.2 Servidores preinstalados

Si se requieren opciones más complejas de streaming, por ejemplo para el uso de aplicaciones, los servicios mencionados previamente están limitados. Ya sea por el hecho de que solo permiten streaming de video, o por el hecho de que dependen de aplicaciones terceras, EULAs (End Users License Agreements) o licencias que restringen su uso.

Para requerimientos más complejos se debe manejar un servidor privado con Adobe Media Server, Red5 o Wowza que permite ajustarse a las necesidades de la aplicación. La instalación y mantenimiento de un servidor o en el caso de Adobe Media Server o Wowza manejar una licencia puede ser complicado. Para estos casos se puede alquilar servidores con estos servicios ya instalados. Dependiendo del caso hay opciones de servidores virtuales y servidores físicos dedicados.

Buscando en el Internet se puede encontrar varios tipos de hosting fácilmente para cualquier de los servicios mencionados.

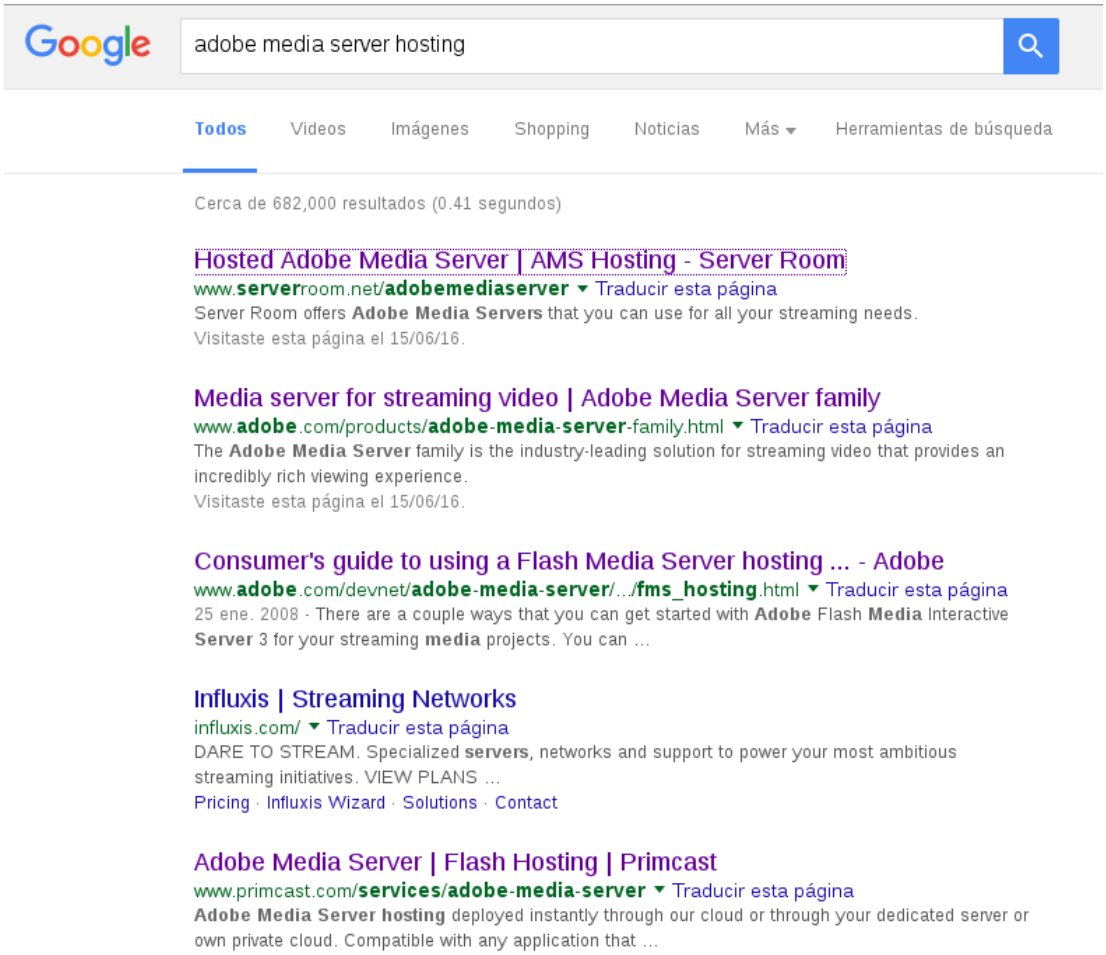

Ilustración 1: Búsqueda de hosting para Adobe Media Server en google. Captura de pantalla.

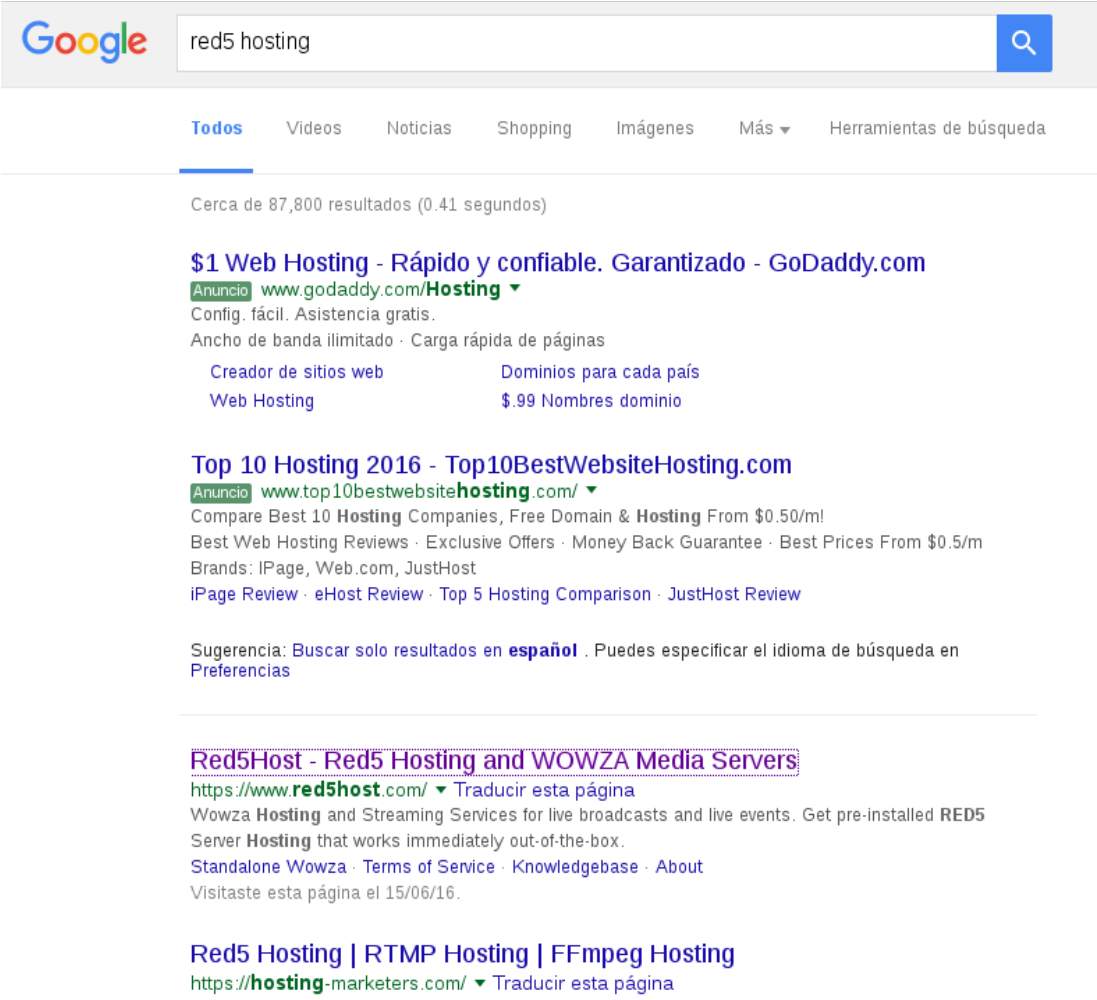

Ilustración 2: Búsqueda de hosting para Red5. Captura de pantalla.

1.3 Servicio en la nube

La demanda de Cloud Services o servicios en la nube está subiendo. Para el 2016 los gastos en infraestructura para servicios en la nube se pronostican cerca de 38 mil millones de dólares y se estima 173 mil millones de dólares para el 2026. En el 2015 Amazon Web Services generó 7.88 mil millones de dólares en su último cuarto (Columbus, 2016).

En el número de proveedores de servicios en la nube va creciendo y entre los nombres más grandes están Amazon, Google y Microsoft. Amazon es el mayor proveedor teniendo cerca de 10 veces más capacidad computacional que su competencia más cercana (Butler, 2015).

Por ser el servicio más popular fue tomada para la propuesta que se desarrolló para la tesis.

1.4 Herramientas para el streaming libre y balanceo de carga aplicados en la investigación

1.4.1 Red5

Red5 es la opción más económica, siendo 100% software libre, código abierto y gratuito. Es un servidor una aplicación que inicialmente intenta proveer los mismos servicios que Adobe Media Server. ("Red5 (media server)", 2015) Tiene una comunidad activa de software libre, en la cual he participado y me han ayudado con temas técnicos para esta tesis. Su código se lo puede ver en GitHub, https://github.com/Red5.

De acuerdo a su página web http://red5.org, Red5 fue creado originalmente para dar soporte a RTMP, pero hoy en día es usado para streaming en vivo incluyendo HLS (HTTP Live Streaming), WebSockets y RTSP (Real Time Streaming Protocol).

Existen empresas profesionales como Red5Pro, con su sitio web https://red5pro.com, que proveen servicios y soporte para Red5, con una versión de pago de Red5. La versión de Red5 Pro requiere una licencia de pago, aunque tiene una versión para desarrolladores.

Red5 tiene embebido un contenedor de Tomcat JEE Web Aplications, usando el framework de Spring. Esta basado en licencia de Apache *License* 2.0..("Red5 (media server)", 2015)

1.4.2 JBOSS/Wildfly

JBoss es un servidor de aplicaciones Java EE utilizando Java. Tiene una Licencia LGPL versión 2.1 lo cual le hace software libre y abierto. Originalmente tenía una versión comunitaria y gratuita llamada simplemente JBoss AS (Aplicaction Server o servidor de aplicaciones) y otra versión de pago por suscripción también conocida como JBoss EAP o JBoss Enterprise Application Platform. A partir de la versión 8 se empezó a llamar WildFly en vez de JBoss AS ("WildFly", 2016).

JBoss EAP mantiene un sistema de versiones un poco más antiguo y conservador que JBoss AS. Recibe soporte directamente de RedHat por lo cual se puede acceder sus servicios en caso de recibir ayuda ("JBoss EAP - Overview", s/f).

JBoss/Wildfly soporta varios tipos de cluster a varios niveles y provee balanceo de carga y soporte para fallas, esto le permite manejar errores sin que los usuarios finales pierdan conectividad. (Mozaffari, 2013, p. 3) Idealmente un cluster de Wildfly/JBoss es visto como una sola instancia, mientras que la redundancia y replicación es transparente para el cliente final (*Ibíd*., p. 4).

Clusters se refiere al uso de múltiples recursos, por ejemplo servidores, como una sola entidad. Los clusters más sencillos se manejan con un balanceador de carga que permite distribuir la carga entre varios servidores, tratando de manejar los peticiones de los clientes sin que pierdan sesiones (*Ibíd*., p. 3).

La combinación de las herramientas Red5 y clusters de JBoss/Wildfly con un balanceo de carga da la posibilidad de prestar un servicio de alta disponibilidad.

Capítulo 2: Alta Disponibilidad

Este capítulo tiene como objetivo el "estudiar el balanceo de carga en la nube". Se basa mayormente en la documentación oficial de Red Hat para JBoss con el documento "*JBoss EAP 6 Clustering, JBoss Enterprise Application Platform 6.1: High Availability, configuration and best practices*". Este documento se puede aplicar para las versiones de JBoss EAP 6 y superiores y también para JBoss AS 7 hasta WildFly 10.

Esta es la única fuente disponible para el desarrollo de este tema. Los puntos más importantes son:

2.1 Arquitectura de alta disponibilidad

Según Babak Mozaffari (2013) existen varios tipos de arquitecturas de alta disponibilidad que se pueden aplicar a JBoss/Wildfly dependiendo de la cantidad de usuarios, carga necesitada y recursos disponibles. JBoss/Wildfly está diseñado para la escalabilidad, crecimiento y alta disponibilidad usando otras herramientas como balanceadores de carga para manejar las distintas sesiones de usuarios (p. 17).

2.2 Balanceo de carga

Un balanceador de carga distribuye las peticiones de los usuarios. Esto se puede hacer por medio de software o hardware. JBoss puede funcionar con varios tipos de balanceadores.

Al manejar varias sesiones de usuario se debe mantener de alguna forma qué información pertenece a quién. Por ejemplo, si un usuario está conectado con una sesión http, digamos que está registrado a un sitio que tiene un carro de compras (p. 5), la información de la sesión se almacena en el servidor y en el lado del cliente, de esta forma cuando el usuario se redirige a otra página dentro del sitio ya se sabe que el usuario está registrado y puede ingresar a la información del carro de compras respectivo.

2.2.1 Balanceo de carga Sticky

En el caso que el balanceador de carga común redirija al usuario a otro nodo, la página web se puede mantener pero el usuario perdería la información de la sesión. Por tanto, tendría que volver a ingresar al sitio y poner la información del carro de compras nuevamente, si es que no se grabó en la base de datos. Por esto el balanceo de carga debe ser Sticky ya que permite una forma de manejo de nodos que es invisible para el usuario final y mantener la sesión.

Balancear la carga no es solo la única forma en la que el usuario es redirigido a otro servidor o nodo. Esto también se puede dar por fallas del lado del servidor. Un sistema de alta disponibilidad no debería perder la información del usuario nunca. Perder la información de sesión puede ser muy costoso, especialmente si la información es sensible. Por ejemplo información de transferencias bancarias.

2.3 Tipos de clusters

El tipo más sencillo de clusters se maneja con al menos 2 servidores. La idea más fácil para replicar un cluster es tener copias de las sesiones en todos los nodos. De esta forma cuando un balanceador de carga cambia a un usuario a otro nodo su sesión se mantiene (Mozaffari, 2013, p. 5).

Ilustración 3: Cluster y replicación de sesión de HTTP. Fuente: Mozaffari, 2013, p. 5

En el gráfico superior de clusters y replicación de sesión de HTTP se tiene como ejemplo a 3 nodos o servidores en los cuales se va a asumir que cada uno tiene una sesión. Las sesiones están representadas con una letra (A, B y C). La replicación en esta forma más sencilla se mantiene copiando cada una de las sesiones a todos los servidores (Mozaffari, 2013, p. 5).

Esta opción está recomendada para clusters pequeños debido a que si se escala horizontalmente se va a mantener una copia de todas las sesiones. Esto se convierte en un problema escalar, ya que el número de sesiones por el número de servidores va a indicar todos los recursos que se están utilizando. Si tenemos un número mayor de servidores no tiene sentido replicar todas las sesiones, porque no se espera que fallen todos los servidores. Esto significa un desperdicio de recursos.

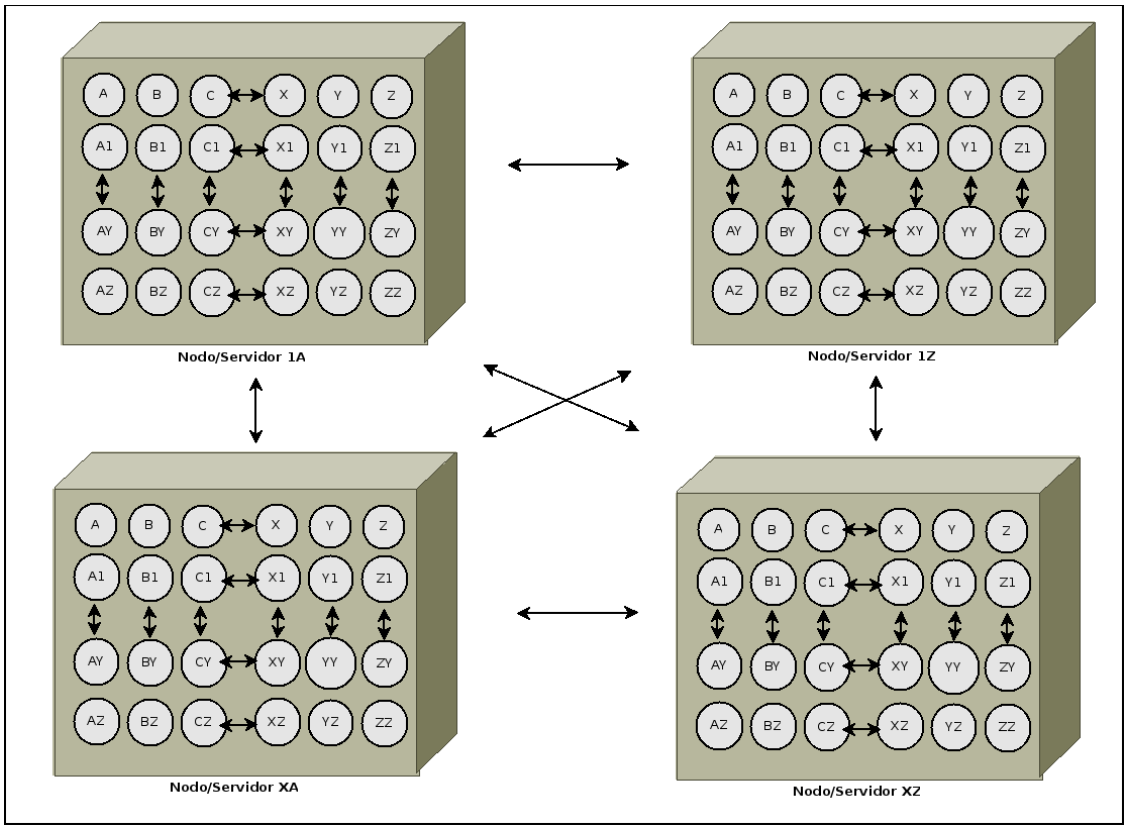

Ilustración 4: Problemas de escalabilidad horizontal en replicación de sesiones en clusters grandes. Fuente: Mozaffari, 2013, p. 6

Como se puede ver en la gráfica superior se necesitarían servidores con muchos recursos para poder sustentar este modelo a una escala más grande. Se vuelve contraproducente que por cada recurso se necesiten muchos más recursos. Sobre todo si estos son recursos que no se están usando.

2.4 Clusters de distribución de sesión para clusters grandes

Para manejar de una manera más eficiente los recursos en escalas más grandes se puede configurar a los nodos de manera distribuida. Bajo el modo de distribución se puede crecer de manera lineal de acuerdo a las cantidades de servidores que se agreguen. Su eficiencia viene a darse que al momento de copiar y replicar la información de las sesiones solo se transmite parte de la información a un número determinado de servidores. De esta forma nos aseguramos que las sesiones están replicadas en algunos servidores pero no en todos. Siendo mucho más eficiente con los recursos.

Por defecto el modo de distribución para JBoss está configurado para 2 nodos, garantizando que la sesión va a estar duplicada. Esto sirve para mantener un respaldo de por lo menos un servidor. Si se pone un número mayor se puede garantizar una menor probabilidad de pérdida de datos de sesión, pero a cambio de usar más recursos por servidor (Mozaffari, 2013, p. 7)

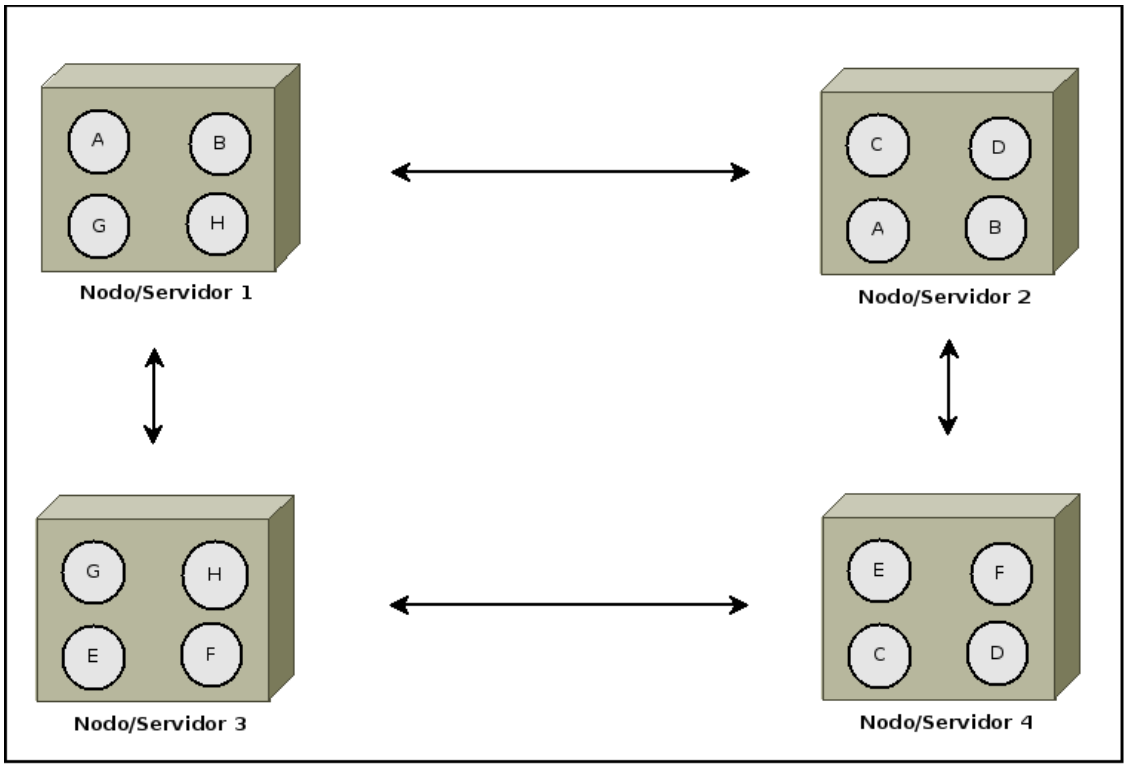

Ilustración 5: Cluster de Distribución de Sesión HTTP para clusters de Grandes Escalas. Fuente: Mozaffari, 2013, p. 7

En la gráfica superior se explica el manejo de 4 nodos con 8 sesiones, manteniendo la distribución para 2 servidores. Si no fuera un sistema distribuido, se tendría 8 sesiones por servidor utilizando muchos más recursos.

2.5 Balanceo de carga en la nube

Lo interesante de balancear carga en la nube es que se puede manejar mucho mejor la escalabilidad. Al ser servidores virtuales se puede crear nodos en cuestiones de segundos o minutos en caso de necesitarse y al no estar utilizados se puede devolver los recursos. Al usar más recursos esto cuesta más pero al no usarlos el gasto se disminuye. Un cluster con servidores físicos no puede devolver los recursos no usados.

2.5.1 Elastic Load Balancing de Amazon Web Services con sesiones Sticky

Amazon provee un balanceador de carga con sesiones Sticky, se llama Elastic Load Balancing. Este balanceador de carga produce una *cookie* o galleta especial la cual permite registrar qué instancia maneja la petición del cliente o usuario. Esta galleta es insertada en los paquetes de las peticiones para así seguir el rastro. Si una instancia de un servidor deja de funcionar o responder, el balanceador de cargas redirige el tráfico hacia otro servidor ("Configure Sticky Sessions for Your Load Balancer - Elastic Load Balancing", s/f).

Las sesiones Sticky solo nos van a servir para la parte web de las aplicaciones en Red5. JBoss no est diseñado directamente para el streaming por lo cual las sesiones de RTMP

creadas en Red5 no se van a replicar a los nodos del cluster. Esto se debe solucionar en la capa de la aplicación.

2.5.2 JGroups

JGroups es un kit de herramientas de mensajería que puede ser usado para la creación de clusters permitiendo la comunicación entre nodos. Maneja la comunicación de nodo a nodo o también de un nodo al cluster entero, la detección de nuevos nodos, y la salida de nodos del cluster. De esta forma los nodos se pueden conectar a través de la redes LAN o WAN a distancia ("JGroups - The JGroups Project", s/f).

JBoss/WildFly ya contiene a JGroups como un sub-sistema. Por defecto el modo de conexión utiliza el puerto 7600, aunque se puede configurar para usar otro puerto. La pila de manejo de información se denomina *stack* y puede manejarse de 3 formas:

- UDP, esta es la opción por defecto. Por medio de un *multicast* se descubren todos los nodos que respondan y que estén configurados adecuadamente. Es perfecto para el auto-escalamiento ya que solo requiere levantar un servidor con la configuración adecuada y es inmediatamente reconocido por el cluster. Por motivos de seguridad Amazon Web Services no permite multicast (Mozaffari, 2013, p. 54).

- TCP, cuando no se puede hacer multicast se puede usar TCP. En el archivo de configuración del JBoss/WildFly se especifican las direcciones IP de los nodos y por medio de un ping se sabe cuáles servidores están funcionando o no. Esta opción requiere que de forma manual se ingresen las direcciones IP de los nodos a los archivos de configuración y no permite una recuperación rápida en el caso de caída de un servidor. No permite un auto-escalamiento (*Ibíd*., p. 54).

- S3Ping es una variación de TCP utilizando el almacenamiento de AWS S3. Utiliza un *bucket* de S3 o cubo de almacenamiento para guardar una lista, que es básicamente un archivo de texto de todos los servidores que están configurados para usar S3Ping. Mediante un ping hacia las direcciones IP de los servidores se sabe si están arriba o no. De esta forma puede usar el auto-escalamiento adentro de Amazon Web Services ("Elastic JBoss AS 7 clustering in AWS using EC2, S3, ELB and Chef - Amazon Web Services Germany", 2013).

2.5.3 Infinispan

Infinispan es un tipo de memoria distribuida para almacenar datos. Puede ser usado como una librería Java o como un servicio independiente de cualquier lenguaje de programación al que se puede acceder remotamente por medio de varios protocolos. Ofrece funcionalidades avanzadas como transacciones, eventos, peticiones y procesamiento distribuido ("Infinispan Homepage - Infinispan", s/f) La capa de Infinispan opera sobre los JGroups para distribuir el cache hacia los nodos.

Luego de estas explicaciones a partir de los siguientes capítulos se iniciará el proceso de construcción del cluster de streaming.

Capítulo 3: Proceso de instalación del cluster

El objetivo de este capítulo es presentar la implementación de aplicaciones de software libre para streaming de alta disponibilidad.

El servicio de EC2 viene del inglés Amazon Elastic Compute Cloud, básicamente es un servicio de máquinas virtuales o servidores que se alquilan en la web. Cada usuario puede alquilar servidores de acuerdo a sus requerimientos incluyendo cantidad de ram, capacidad de procesamiento, almacenamiento en disco y hasta interfaces de red (“Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS”, s/f).

El proceso seguido para la creación del cluster es el siguiente:

3.1 Instalación del servidor en EC2 de Amazon Web Services

Lo primero que vamos a hacer es instalar un servidor base con Linux para que pueda correr JBoss/WildFly en Amazon Web Services. El servicio de AWS que permite crear servidores virtuales se llama EC2.

Para ingresar a la consola de Amazon Web Services debemos ingresar a la consola de AWS, https://console.amazon.com. Al ingresar los primero que vemos es el Dashboard o panel central de Amazon donde podemos elegir los distintos servicios.

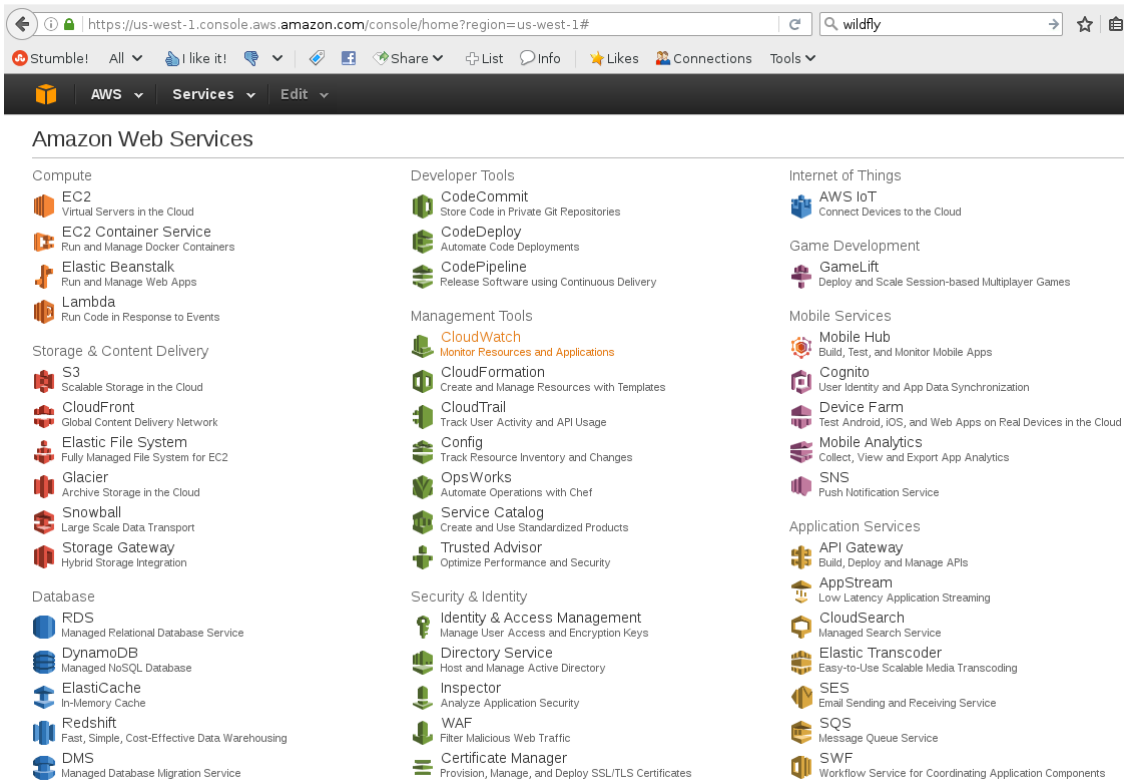

Ilustración 6: Ejemplo de panel de servicios de Amazon Web Services. Captura de pantalla.

Entre las primeras opciones podemos ver EC2, *Virtual Servers in the Cloud.* Aquí es donde vamos a crear a nuestros servidores. Amazon tiene varios centros de cómputo a lo largo del planeta, en distintos países y continentes. Los precios de uso por servidor varían en cada región. Se puede crear servidores de acuerdo a la región que uno esté o en la que quisiera mostrarse para que haya menos saltos entre las redes. Esto puede ser bien útil para casos de streaming en los cuales cada salto de red afecta a la calidad de la señal.

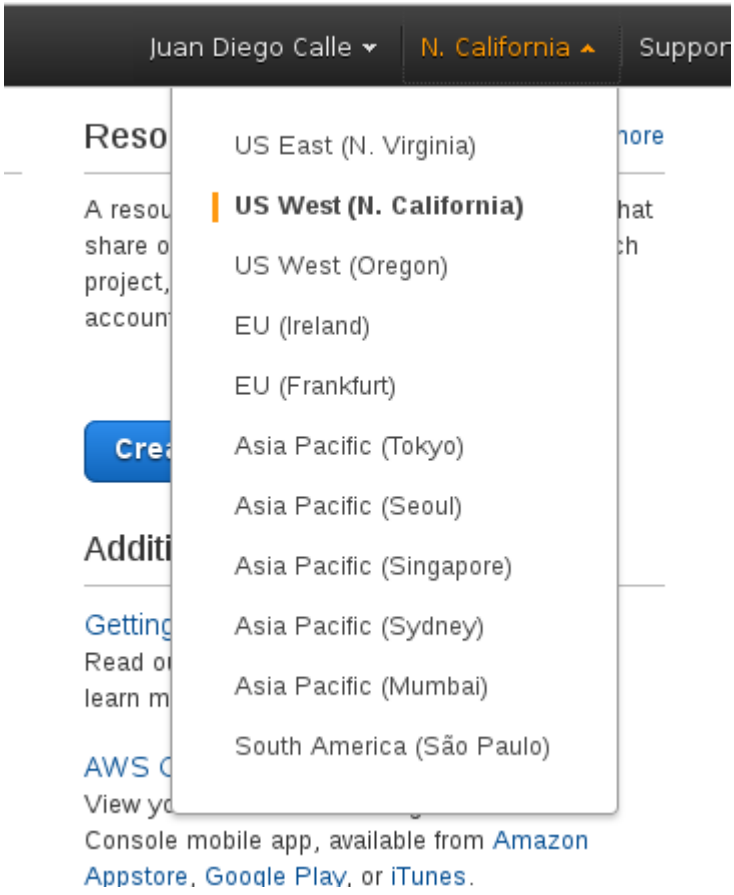

Ilustración 7: Ejemplo de regiones disponibles de Amazon. Captura de pantallas.

Para el caso de Ecuador sería beneficioso usar los servidores en Brasil que en teoría están más cerca que cualquier otro pero el costo por hora es mayor. Por ejemplo en la página oficial de AWS https://aws.amazon.com/ec2/pricing/ se puede comparar los precios. En una instancia de Linux ts.medium con 2 CPU virtuales y con 4 GiB de RAM costaría a la fecha de desarrollo de este documento $0.052 la hora en los Servidores de Virginia del Norte o US East(N. Virginia), mientras que un servidor en Brasil de iguales capacidades costaría $0.108 la hora. El costo es más del doble.

Ahora si vamos al panel de EC2 o EC2 Dashboard podemos crear instancias, manejar el balanceo de carga y hasta crear los grupos para la auto-escalabilidad.

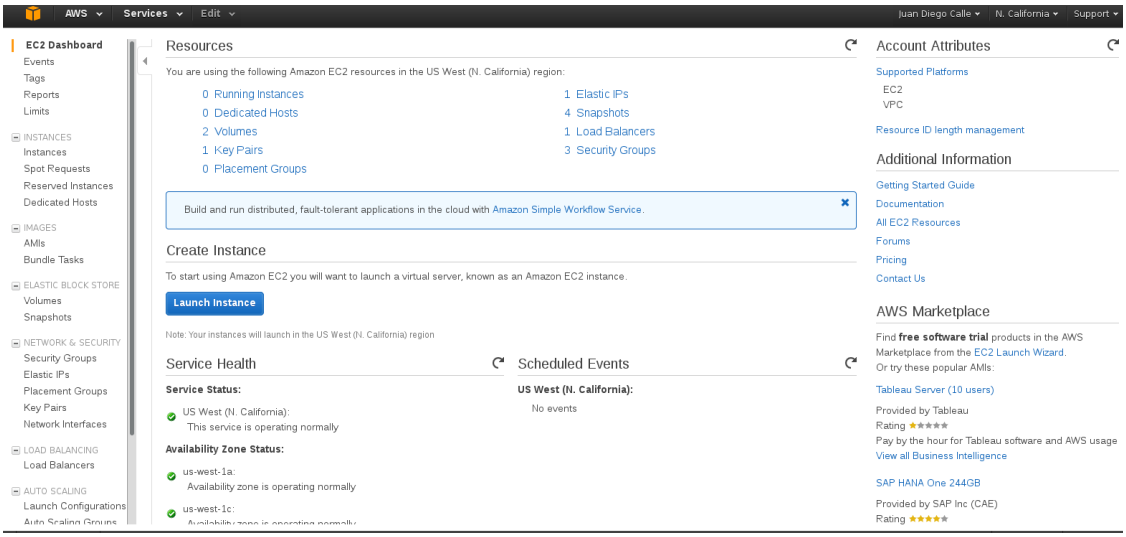

Ilustración 8: Panel de EC2 o EC2 Dashboard. Captura de Pantalla.

Ahora podemos irnos a donde dice *Instances* o Instancias y podemos ver todas las instancias creadas. En esta interface es donde podemos detectar si hay algún servidor caído a nivel de sistema operativo, también podemos crear o eliminar nuevas Instancias.

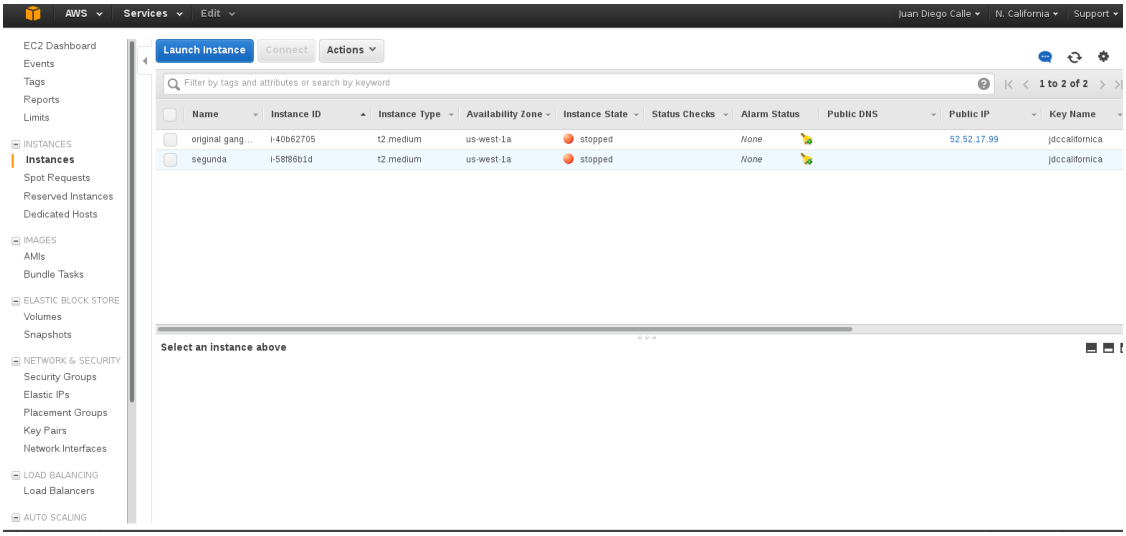

Ilustración 9: Instancias de EC2. Captura de Pantalla.

Entonces el siguiente paso es lanzar una instancia, por lo cual vamos al botón 

o “Lanzar o Crear una Instancia”. Una vea que iniciamos una instancia nos vamos a un *Wizard* o asistente de instalación en el cual podemos usar un AMI. AMI viene de las siglas en inglés Amazon Machine Images.

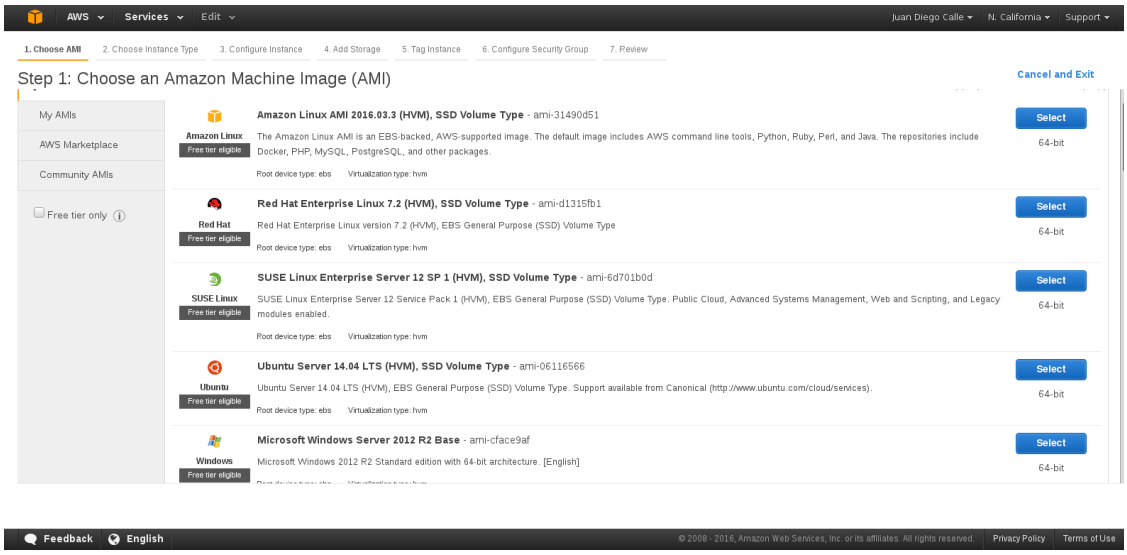

Ilustración 10: Creación de Instancias EC2 paso 1: AMIs o Amazon Machine Images. Captura de pantalla.

Los AMIs son imágenes de servidores con distintos sistemas operativos y en algunos casos con programas y servicios pre configurados. Se tiene varios tipos de Linux como se puede ver, desde Amazon Linux una versión de Linux basada en Red Hat/Centos creada y mantenida por Amazon, hasta Red Hat y Ubuntu. También se puede usar varios tipos de Windows. Cabe recalcar que fuera de las distribuciones gratuitas de Linux al usar Red Hat o Windows incurren alzas de costos para usar las distintas licencias de Microsoft Windows Server o Red Hat por encima del costo usual de cada máquina virtual.

Para la prueba de nuestro cluster vamos a utilizar Amazon Linux AMI debido a que es basada en Red Hat/Centos y ya he tenido experiencia con esta distribución.

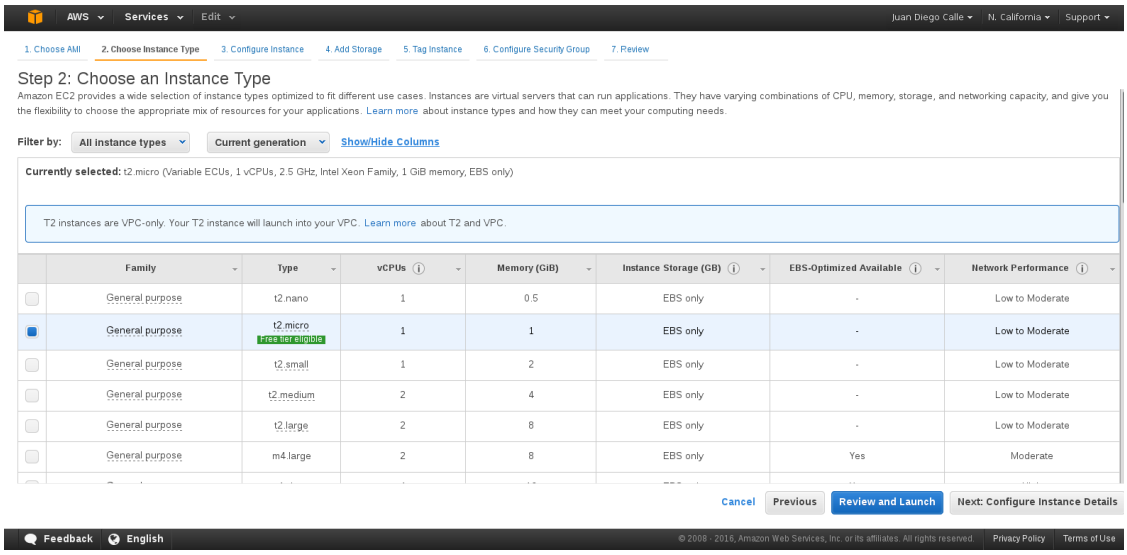

Ilustración 11: Creación de Instancias EC2 Paso 2: Tipos de instancia. Captura de pantalla.

Aquí podemos elegir el tamaño y tipo de nuestra instancia, para nuestro caso vamos a usar 4GiB de RAM asumiendo que el sistema operativo usuaría 2 GiB de RAM y otros 2GiB de RAM para el WildFly. En las pruebas que se realizaron para este caso el sistema operativo nunca usó más de 1 GiB de RAM, por lo cual tenemos más que suficiente espacio para WildFly.

En este caso vamos a elegir un instancia de tipo t2.medium con dos CPUs virtuales o vCPUs, más 4 GiB de RAM. No recomiendo usar ninguna instancia menor a esto porque t2.medium es la mínima instancia para poder utilizar 2 interfaces de red. Más adelante se va aclarar porque se necesitarían 2 interfaces de red.

Podemos ir al siguiente paso donde el asistente de instalación nos guiará. Es donde podemos aumentar el almacenamiento o capacidad de disco de nuestro servidor, sin embargo vamos a dejar las opciones por defecto. Para este caso se va a poner en Review and Launch 

.

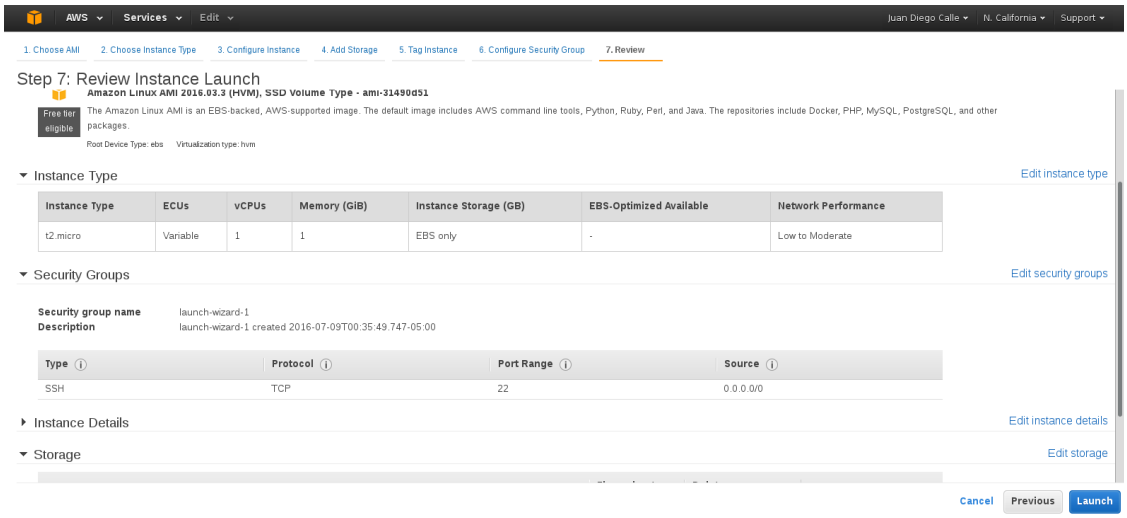

Ilustración 12: Creación de Instancias EC2 paso 7: Revisar lanzamiento de Instancia. Captura de pantalla.

En este punto se puede desplegar nuestro servidor seleccionando el botón Launch. Lo único que nos falta es modificar o crear los grupos de seguridad o *Security Groups* para acceder a los distintos tipos de puertos que necesitan nuestras aplicaciones. Por ejemplo los puertos 1935 de RTMP o el 8080 usado por defecto en WildFly.

Entonces se debe dar clic en *Edit security groups* o editar grupos de seguridad y dar acceso a los puertos necesarios. No hace falta crear un grupo de seguridad para cada servidor, se recomienda reutilizar los grupos de seguridad para servidores con las mismas configuraciones.

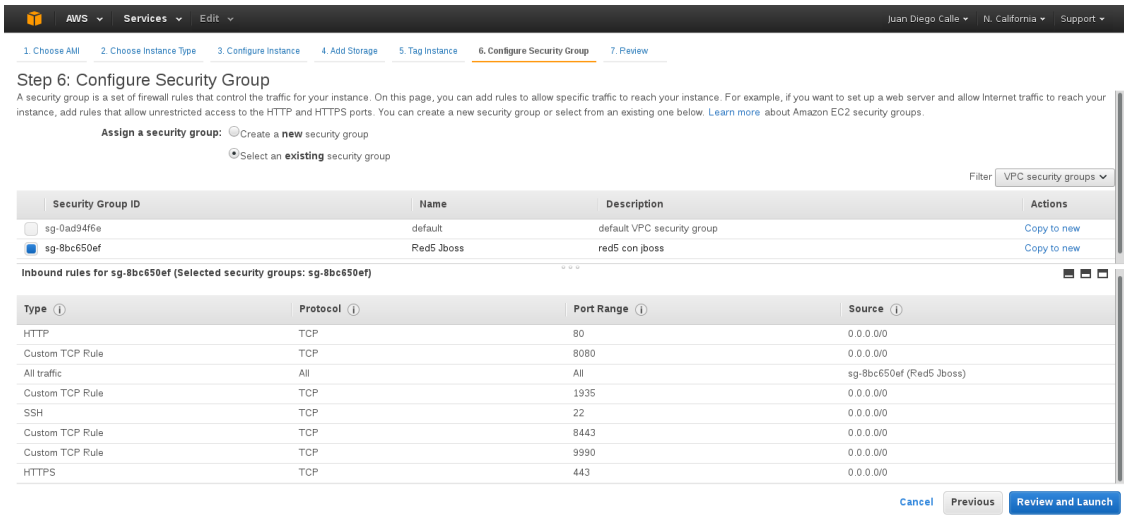

Ilustración 13: Creación de Instancias EC2 Paso 6: Grupos de seguridad con ejemplo. Captura de pantalla.

Al ir a los grupos de seguridad retrocedemos un paso. Para nuestro caso, se permitió el acceso al protocolo TCP del puerto 22 para poder ingresar con un *shell* seguro o SSH, al puerto 1935 donde opera RTMP por defecto y a los puertos web 80, 443, 8080, 8443 usados por los servidores web y JBoss/WildFly. Originalmente iba a usar *proxies* inversos para acceder por los puertos 80 y 443 hacia el JBoss/WildFly, pero no son necesarios para este trabajo. Los puertos 443 y 8443 sirven para utilizar SSL con certificados que los dejo como prueba de concepto, aunque no serán utilizados (Ver Graf…). El puerto 9990 es la consola de administración de JBoss/WildFly, realmente este puerto no es necesario y es preferible que no se tenga acceso desde afuera ya que puede ser peligroso dejar la interfaz administrativa a disposición de cualquier usuario.

Los servidores que se han probado no están dentro de ningún servicio de indexación como Google lo que haría muy difícil que cualquier posible atacante supiera que existen. Además, las instancias han sido levantadas por algunas horas y solo en algunos casos hasta un poco más de un día, por lo cual el riesgo de que sean vulneradas es mínimo. En la mayoría de los casos se han las instancias cuando se acabó de usarlas.

Existe una regla *custom* o personalizada que permite a todo el tráfico de la red “sg-8bc650ef (Red5 Jboss)” ingresar a cualquier puerto. Esta red “sg-8bc650ef (Red5 Jboss)” es nuestra red interna California de Norte donde operarán todos los servidores con JBoss/WildFly. Esta regla podría ser más segura si se le separa en 2 reglas, un acceso a toda la red al puerto 7600 y otra regla para permitir el *ping* entre los servidores de la red. Como son servidores de prueba s prefirió dar preferencia hacia la conectividad antes que la seguridad. En servidores de producción esto tiene que cambiar.

Volvemos a Review y ponemos Launch o lanzar.

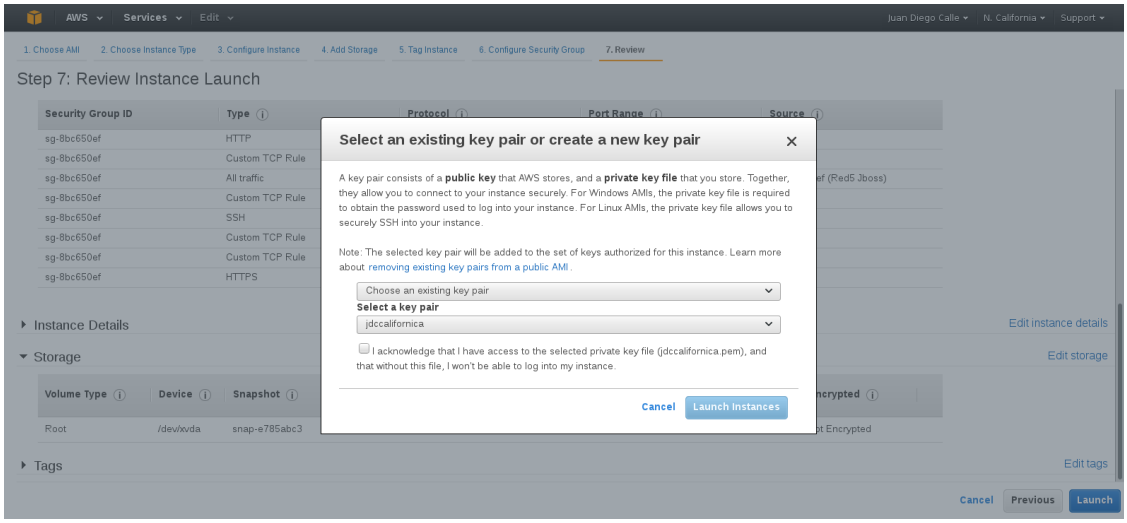

Ilustración 14: Creación de Instancias EC2: Seleccionar llave. Captura de pantalla.

Si no se tiene una llave de acceso se puede crear una. Para este ejemplo ya teníamos creada y descargada una llave. Seleccionamos el cuadro inferior para decir que si

tenemos acceso a la llave ya mencionada. Esta llave nos va a permitir ingresar a nuestros servidores vía SSH.

Luego se va Launch Instances 

. Volvemos al panel de EC2 donde

vamos a ver al estado de nuestra instancia como *Pending* o pendiente. Se puede demorar unos segundos y hasta minutos en base a nuestra experiencia personal.

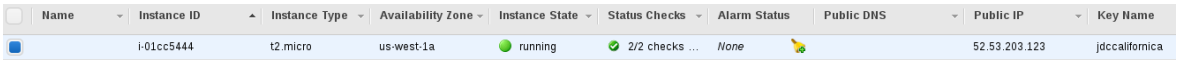

Ilustración 15: Ejemplo de instancia corriendo. Captura de pantalla.

Una vez ya iniciada podemos seleccionamos la instancia y damos clic en el botón *Connect* o Conectar. Donde nos muestra un ejemplo de cómo podemos conectarnos al servidor con nuestra llave.

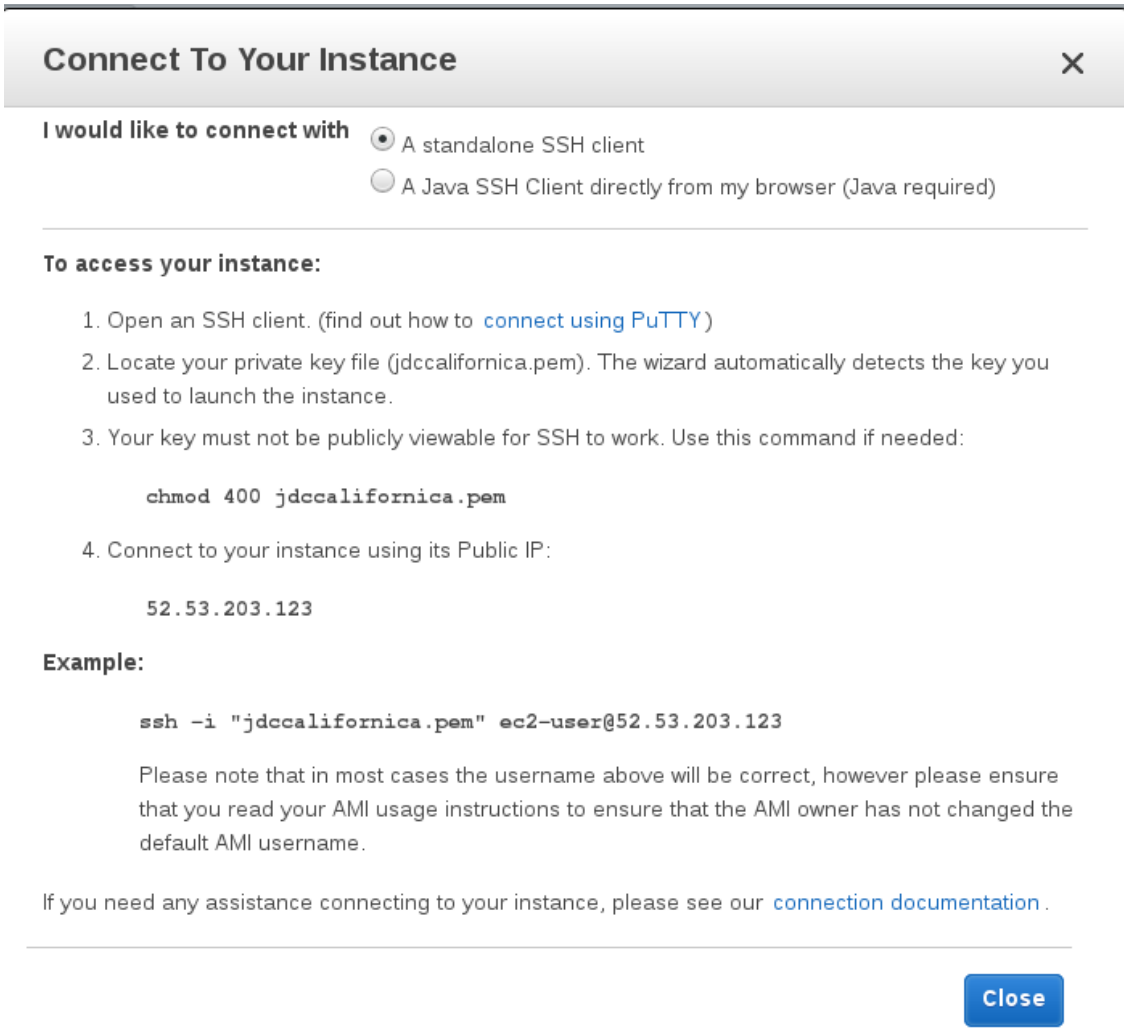

Ilustración 16: Ejemplo de conexión vía SSH a nuestro servidor. Captura de pantalla.

Si no se tiene un cliente SSH desde Windows se recomienda usar Putty descargándolo de http://www.putty.org/, o se puede usar la versión web de Java SSH Client del Graf….anterior.

Para el ejemplo si se está usando una máquina con Linux, se debe ir a la carpeta donde se tiene la llave privada y luego se ejecuta los comandos mencionados.

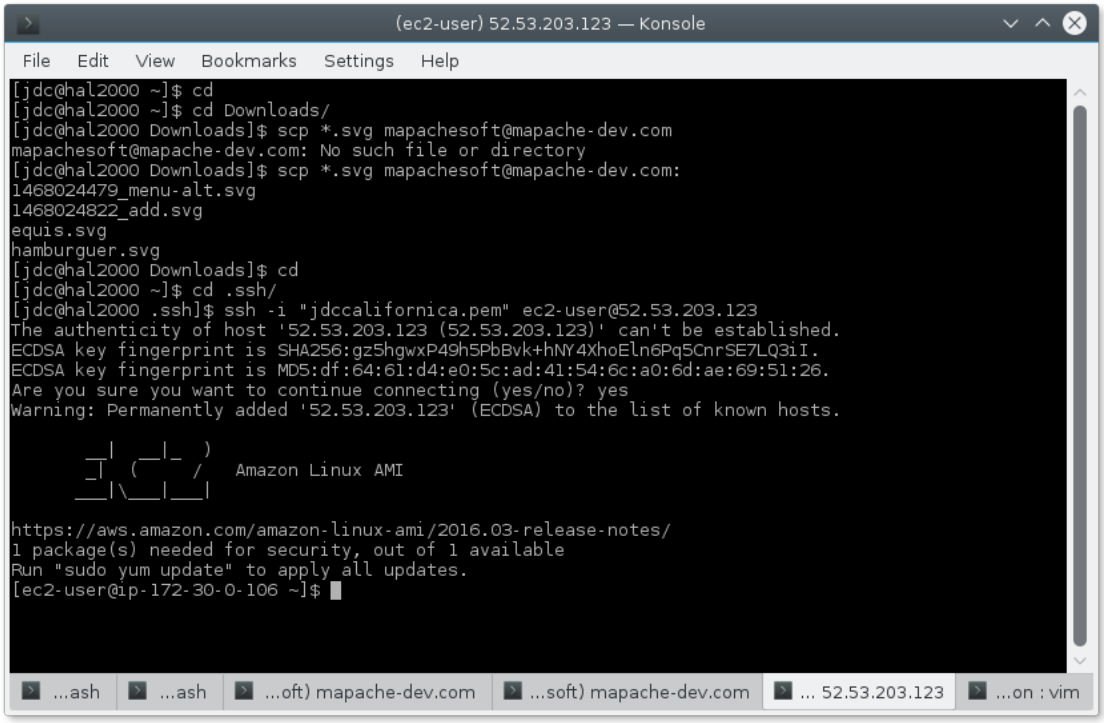

Ilustración 17: Primer acceso vía SSH así mi nuevo servidor. Captura de pantalla.

En la imagen superior se puede apreciar como es el primer ingreso hacia el servidor. La primera vez que ingresamos se guarda la “huella digital” del servidor para poder evitar ataques en los cuales alguien pueda falsear la dirección IP del servidor. Con esto ya podemos instalar lo que queramos dentro del servidor.

3.2 Descarga e Instalación de WildFly 10.0.0

Para la implementación de nuestra aplicación vamos a usar WildFly 10 porque es la última versión estable de WildFly. Se puede apreciar todos los cambios de esta versión en el siguiente link http://wildfly.org/news/2016/01/29/WildFly10-Released/.

Ahora se requiere ir a la página oficial de WildFly http://wildfly.org/downloads/

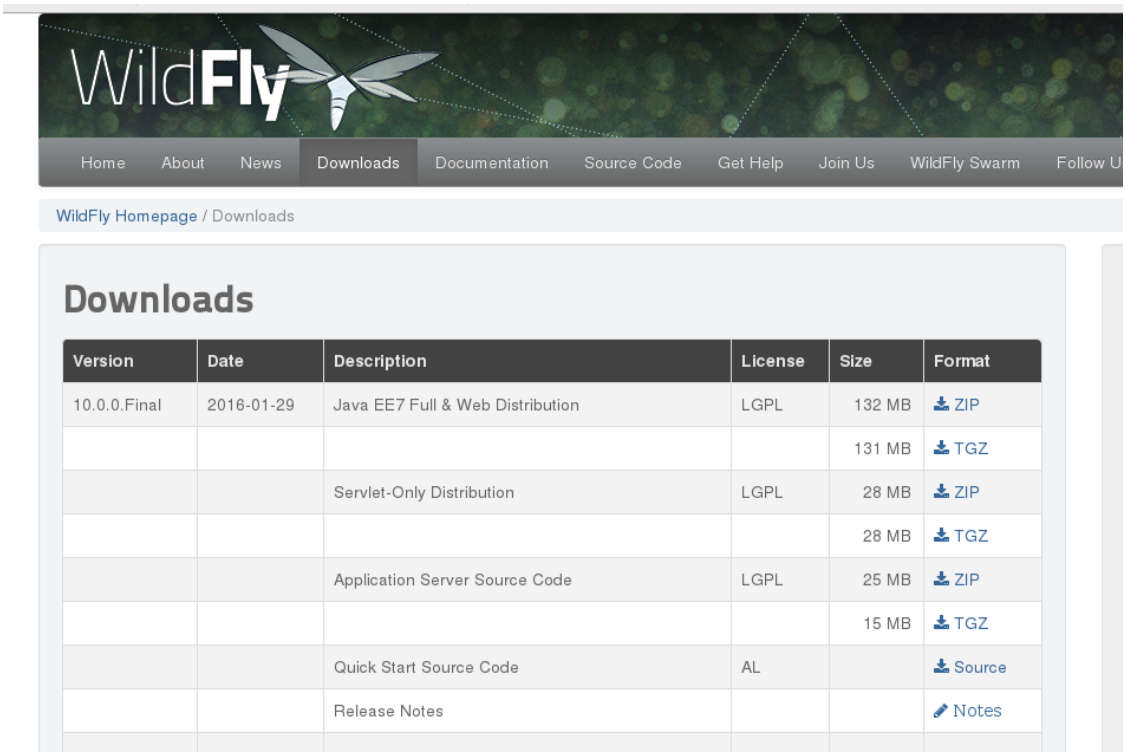

Ilustración 18: Descarga de WildFly en la página oficial. Captura de pantalla.

Vamos a bajar la versión 10.0.0.final en la cual la descripción dice Java EE7 Full & Web Distribution. Viene en 2 formatos Zip o Tgz que son archivos comprimidos que contienen al servidor completo hasta con archivos de configuración de ejemplo.

En este caso se va a descargar el WildFly 10 directamente al servidor, con el comando WGET.

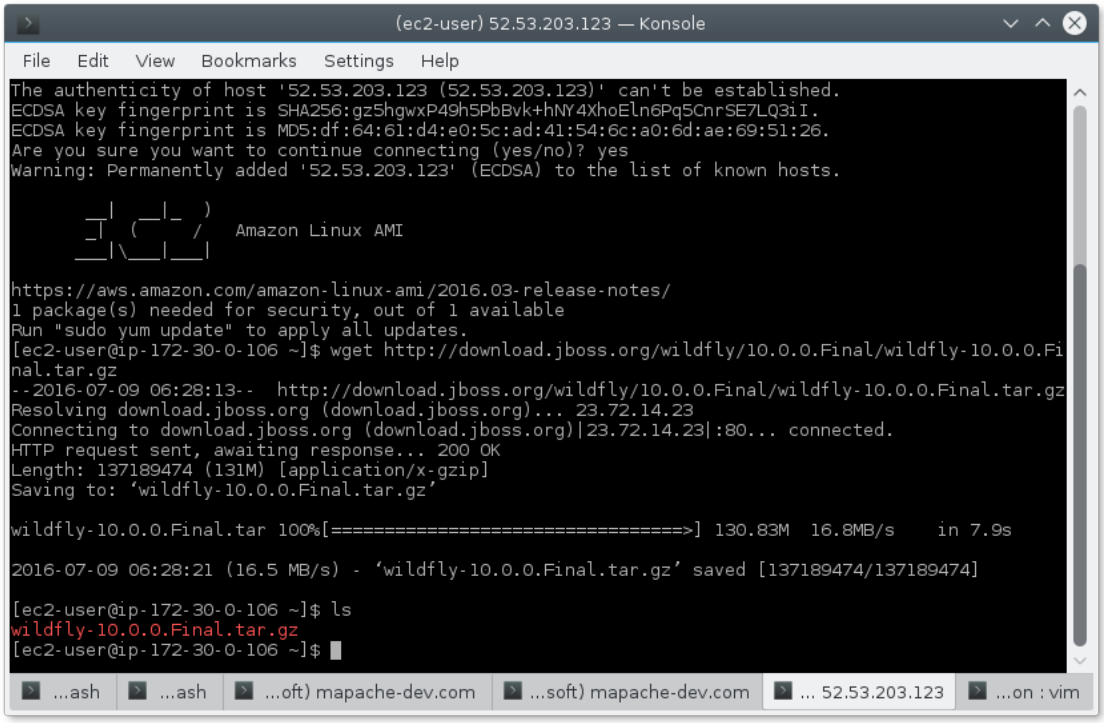

Ilustración 19: Descarga de WildFly 10.0.0.Final en el servidor. Captura de pantalla.

Se acostumbra a descomprimir el archivo en la carpeta "/opt". Por lo cual se debe copiar al archivo comprimido a esa carpeta y luego descomprimirlo con el comando "tar".

La siguiente parte me base en el tutorial "How To Install Wildfly as a Service on Linux" de http://developer-should-know.com/post/112230363742/how-to-install-wildfly-as-a-service-on-linux. Está basado en WildFly 9.0.1.Final y le vamos a adaptar para WildFly 10.

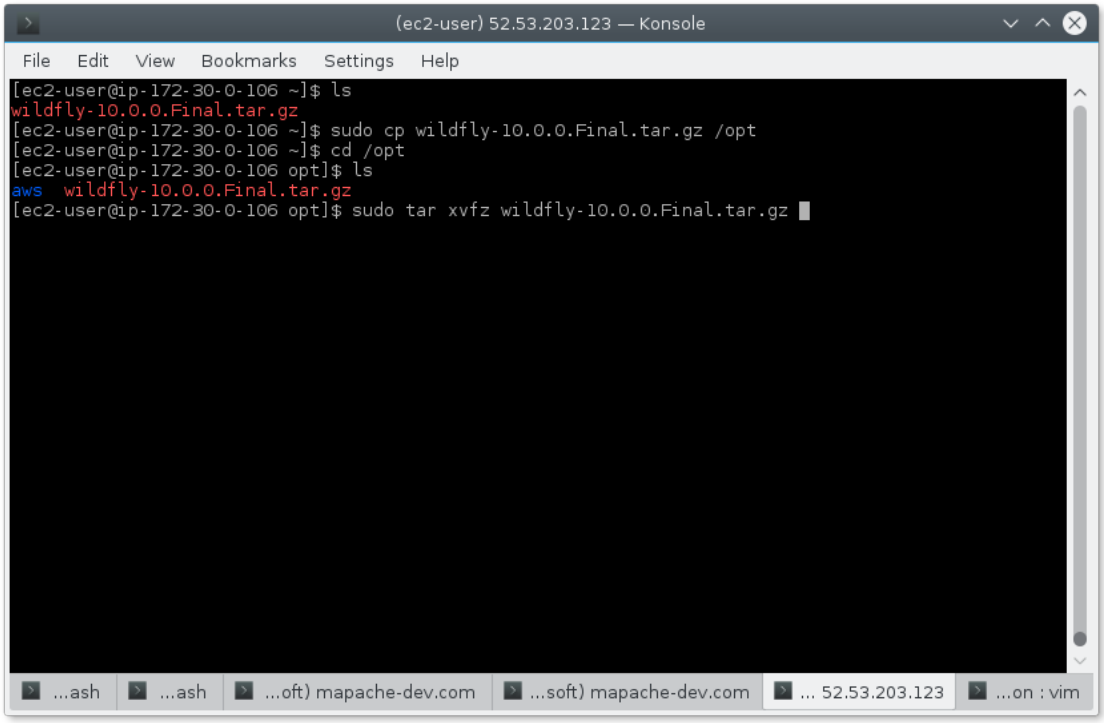

Ilustración 20: Descomprimiendo WildFly 10.0.0.Final. Captura de pantalla.

Para copiar y descomprimir archivos en la carpeta “/opt” se necesitan permisos de super usuario por lo cual uso “sudo”, y para descomprimir un archivo tar.gz se utilizan el comando “tar” con las opciones “xz”. Para más indicaciones se puede leer el manual de “tar”.

Una vez descomprimido el archivo vamos crear un soft link de la carpeta “/opt/wildfly-10.0.0.Final” hacia “/opt/wildfly”. Esto se hace para mantener coherencia y en el caso de actualizar solo redirigimos el nuevo wildfly hacia la carpeta “/opt/wildfly” manteniendo el mismo esquema. El comando para hacer lo antes mencionado es el siguiente:

```
$ sudo ln -s /opt/wildfly-10.0.0.Final /opt/wildfly
```

El siguiente paso es crear un usuario para Wildfly que le vamos a llamar “wildfly” con el HOME de ese usuario en “/opt/wildfly”. Después de crear el usuario es necesario asignar los permisos necesarios a esa carpeta. Esto lo podemos hacer como el súper usuario “root” o con el comando “sudo”.

```
#  sudo su  -

# useradd wildfly -d /opt/wildfly

# chown -R wildfly:wildfly /opt/wildfly
```

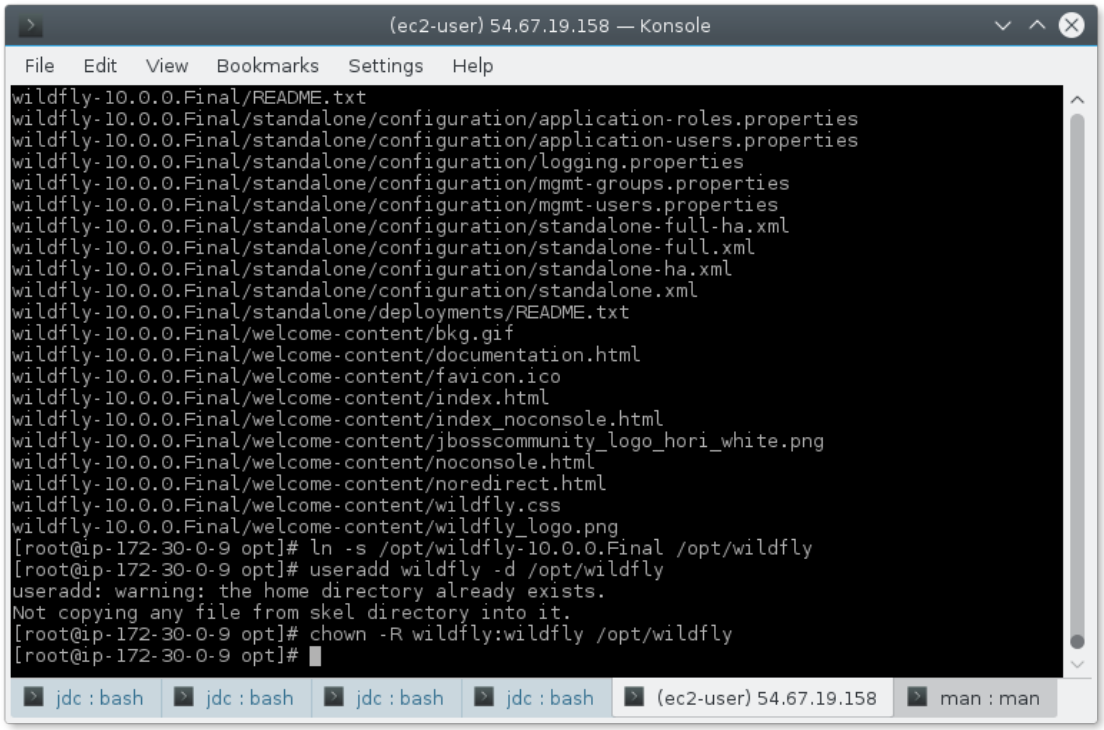

Ilustración 21: Creación de usuario wildfly. Captura de pantalla.

En las versiones de nuevas de WildFly toda la documentación y archivos de configuración vienen en la carpeta “docs”. Debido a que Amazon Linux AMI está basado en Red Hat /CentOs necesitamos los scripts de inicio bajo la carpeta init.d, el tipo de sistema de arranque de Red Hat. El archivo “wildfly.conf” tiene unas variables por defecto que se necesitan para arrancar el WildFly como servicio, por lo cual le vamos a copiar a la carpeta “/etc/default” desde “/opt/wildfly-10.0.0.Final/docs/contrib/scripts/init.d/wildfly.conf”. (“How To Install WildFly as a Service on Linux”, s/f)

Vamos a utilizar Java 8, por lo cual recomiendo instalar la última versión.

```
# yum install java-1.8.0-openjdk
```

Después debemos editar el archivo wildfly.conf para que pueda usar el home.

```
JBOSS_HOME="/opt/wildfly"

JBOSS_USER=wildfly

JBOSS_CONFIG=standalone-full-ha.xml

JBOSS_CONSOLE_LOG="/var/log/wildfly/console.log"
```

JBOSS_HOME, indica donde está instalado el servidor WildFly.

JBOSS_USER, el usuario con el cual va a correr el servicio de WildFly.

JBOSS_CONFIG, esta variable indica que archivo de configuración va a utilizar al intentar levantar el servicio. Como estamos tratando de crear un cluster, en WildFly vienen 2 archivos de configuración por defecto para cluster: standalone-ha.xml y standalone-full-ha.xml. El archivo standalone-full-ha.xml es similar standalone-ha.xml pero tiene incluida la configuración para colas.

JBOSS_CONSOLE_LOG, con esto se puede indicar donde se van a almacenar los logs de WildFly. (“How To Install WildFly as a Service on Linux”, s/f)

El siguiente paso es instalar el script de arranque de WildFly que viene de la carpeta “/opt/wildfly-10.0.0.Final/docs/contrib/scripts/init.d/wildfly-init-redhat.sh”, copiando a la carpeta /etc/init.d/ bajo el nombre wildfly. Con esto podemos el servicio wildfly para que arranque al iniciar.

Ejemplo:

```
# chkconfig --add wildfly

# chkconfig wildfly on
```

Creamos la carpeta “/var/log/wildfly” y le damos permisos para que el servicio pueda escribir los logs.

```
# mkdir -p /var/log/wildfly

# chown -R wildfly:wildfly /var/log/wildfly
```

Y ahora ya podemos arrancar el servicio de wildfly.

```
# service wildfly start
```

Si no hay ningún error el servidor arrancaría sin problemas. Se puede revisar los logs para ver si arrancó o si tuvo un problema. Cabe recalcar que el servicio se demora en arrancar.

A pesar que los puertos del grupo de seguridad están abiertos no va a ser posible ver el puerto 8080 desde afuera. Una forma de asegurarnos que esté funcionando es usando el comando telnet desde el servidor en Amazon.

```
# telnet localhost 8080
```

También se puede revisar el estado del servidor.

Ilustración 22: Estado del servicio widfly cuando esta corriendo. Captura de pantalla.

Una vez listo el servidor se puede iniciar el proceso de configuración y adaptación de Red5 para poder ser usado en Wildfly.

Capítulo 4: Configuración de Red5 con WildFly para usar como cluster

Para completar la implementación del software libre para streaming de alta disponibilidad es necesario realizar algunos ajustes en las configuraciones de Wildfly para que se pueda acceder al cluster desde afuera, y para que los nodos puedan hablarse entre ellos.

4.1 Configuración con Wildfly

4.1.1 Permitir acceso externo a WildFly

Este paso no es obligatorio si no se requiere acceso externo. Si se va a manejar en producción se puede utilizar un proxy reverso, ya que es mucho más seguro no exponer los puertos de JBoss / WildFly directamente. Además, si se utiliza un balanceador de carga el acceso externo no es necesario ya que éste va a manejar las conexiones.

Como estamos haciendo pruebas sobre un cluster es preferible poder revisar desde afuera, por tanto el balanceador de carga está configurado para usar el puerto 8080 y no se va a instalar dentro de este trabajo un proxy reverso.

Es ahí cuando se necesita modificar la interfaz pública y externa para poder recibir peticiones al puerto 8080 desde afuera del servidor. En el archivo de configuración "standalone-full-ha.xml" de nuestro WildFly vamos a modificar la interfaz *public* o pública.

Se busca la interfaz que dice *public*. En el servidor se da acceso a esta interfaz a la tarjeta de red externa. En el caso de prueba es el servidor eth0, por lo que se remueve lo que dice "<inet-address value="${jboss.bind.address:127.0.0.1}"/>" y se cambia por "<nic name="eth0"/>".

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
    </interface>
    <interface name="public">
        <nic name="eth0"/>
    </interface>
    <interface name="private">
        <inet-address value="${jboss.bind.address.private:127.0.0.1}"/>
    </interface>
    <interface name="unsecure">
        <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
    </interface>
</interfaces>
```

Ilustración 23: Modificación de interfaz pública de standalone-full-ha.xml. Captura de pantalla.

Después de hacer el cambio se debe reiniciar el servidor.

```
# service wildfly restart
```

Con esto ya se podría acceder a nuestro servidor desde afuera.

4.1.2 S3Ping

Para permitir la conexión del servidor hacia la cuenta de Amazon y así poder acceder al almacenamiento de Amazon S3 se debe crear un usuario y darle acceso a ese bucket.

El comando puede crear el usuario desde esta línea:

```
$ aws iam create-user --user-name jboss

$ aws s3api create-bucket --bucket wildfly10clusterconfig --create-bucket-configuration '{ "LocationConstraint":"us-west-1" }'
```

Se crea un usuario JBoss para nuestra cuenta. Después se establece un bucket con el nombre “wildfly10clusterconfig” para acceso a la región “us-west-1” para el Norte de California.

El siguiente paso es dar los permisos necesarios al bucket con las políticas necesarias.

```
{
"Id":"Policy1467063181374",
"Version":"2012-10-17",
"Statement":[
  {
    "Sid": "Stmt1467063070343",
    "Action": [
      "s3:ListAllMyBuckets"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:s3:::*"
  },
  {
    "Sid": "Stmt1467063140116",
    "Action": "s3:*",
    "Effect": "Allow",
    "Resource": [
      "arn:aws:s3:::wildfly10clusterconfig",
      "arn:aws:s3:::wildfly10clusterconfig/*"
    ]
  }
 ]
}
```

Ilustración 24: Ejemplo de políticas para acceso al bucket de Amazon S3(“Elastic JBoss AS 7 clustering in AWS using EC2, S3, ELB and Chef - Amazon Web Services Germany”, 2013)

Con el ejemplo anterior creamos un archivo con estas políticas que se va a llamar:

s3_permissions.json y lo vamos a guardar en la carpeta /tmp.

```
$ aws iam put-user-policy --user-name jboss --policy-name jbosspolicy --policy-document file:///tmp/s3_permissions.json
```

Se puede verificar que exista en la página de Amazon.(“Elastic JBoss AS 7 clustering in AWS using EC2, S3, ELB and Chef - Amazon Web Services Germany”, 2013)

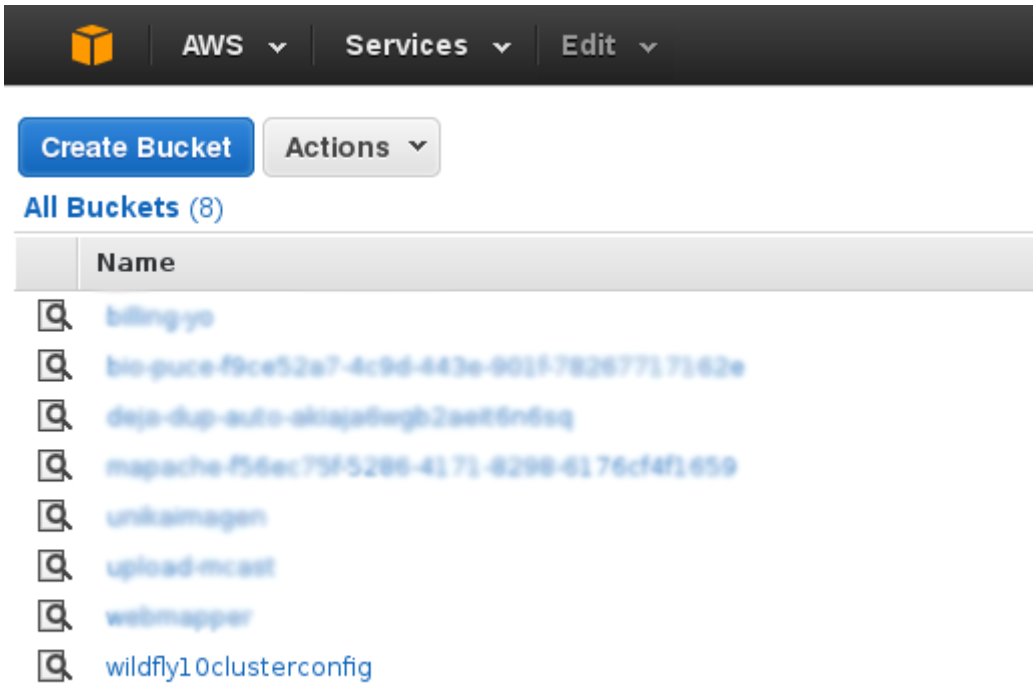

Ilustración 25: Buckets de Amazon S3. Captura de pantalla.

En el sub-sistema de JGroups en el archivo de configuración de WildFly se debe agregar el stack para S3Ping. Se debe buscar la línea que dice:

<subsystem xmlns="urn:jboss:domain:jgroups:4.0">

Ahora se debe crear el stack de s3ping:Se puede v

```
<stack name="s3ping">
                <transport type="TCP" socket-binding="jgroups-tcp"/>
                <protocol type="S3_PING">
                   <property name="access_key"> llave_de_accesso</property>
                                                                                <property
name="secret_access_key">llave_secreta_de_accesso</property>
                   <property name="location">
                      wildfly10clusterconfig
                   </property>
                   <property name="timeout">
                      60000
                   </property>
                </protocol>
                <protocol type="MERGE3"/>
                <protocol type="FD_SOCK" socket-binding="jgroups-tcp-fd"/>
                <protocol type="FD"/>
                <protocol type="VERIFY_SUSPECT"/>
                <protocol type="BARRIER"/>
                <protocol type="pbcast.NAKACK"/>
                <protocol type="UNICAST3"/>
                <protocol type="pbcast.STABLE"/>
                <protocol type="pbcast.GMS"/>
                <protocol type="UFC"/>
                <protocol type="MFC"/>
                <protocol type="FRAG2"/>
             </stack>
```

Ilustración 26: Configuración de stack de s3ping ("Elastic JBoss AS 7 clustering in AWS using EC2, S3, ELB and Chef - Amazon Web Services Germany", 2013)

Se debe utilizar las llaves generadas por Amazon para acceder bajo las propiedades que dicen llaves de acceso y llaves de acceso privadas. En algunos casos se van a encontrar "prefix" para la propiedad del nombre del *Bucket,* en las pruebas generadas dio algunos problemas, es preferible usar "location"

Ahora es necesario decirle al sub-sistema de JGroups que utilice el stack que se acaba de crear. Bajo la línea <channel name="ee" stack="udp"/>, se debe cambiar de udp a s3ping. Ejemplo: <channel nameSe puede v="ee" stack="s3ping"/>.

Los nodos por defecto están configurados con la interfaz privada.

```
</interfaces>
<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">
    <socket-binding name="management-http" interface="management" port="${jboss.management.http.port:9990}"/>
    <socket-binding name="management-https" interface="management" port="${jboss.management.https.port:9993}"/>
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="iiop" interface="unsecure" port="3528"/>
    <socket-binding name="iiop-ssl" interface="unsecure" port="3529"/>
    <socket-binding name="jgroups-mping" interface="private" port="0" multicast-address="${jboss.default.multicast.address:230.0
    <socket-binding name="jgroups-tcp" interface="private" port="7600"/>
    <socket-binding name="jgroups-tcp-fd" interface="private" port="57600"/>
    <socket-binding name="jgroups-udp" interface="private" port="55200" multicast-address="${jboss.default.multicast.address:230
    <socket-binding name="jgroups-udp-fd" interface="private" port="54200"/>
    <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105" multicast-port="23364"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
        <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
```

Ilustración 27: Configuración por defecto de jgroups-tcp. Captura de pantalla

Como se ve puede ver en la ilustración 27 en el ejemplo jgroups-tcp utiliza una interfaz privada por lo que no va a poder ser accedida entre los nodos. En ese momento hay básicamente 2 opciones: o se cambia la interfaz privada para poder acceder a una interfaz de red, o se cambia a una interfaz de jgroups-tcp a una pública. Podría ser recomendable usar una segunda interfaz de red para conectarse a la interfaz privada. Si se tiene una red privada podría manejarse de otra forma. Por motivos prácticos se cambia a la interfaz pública.

```
</interfaces>
<socket-binding-group name="standard-sockets" default-interface="public" port-offset="${jboss.socket.binding.port-offset:0}">
    <socket-binding name="management-http" interface="management" port="${jboss.management.http.port:9990}"/>
    <socket-binding name="management-https" interface="management" port="${jboss.management.https.port:9993}"/>
    <socket-binding name="ajp" port="${jboss.ajp.port:8009}"/>
    <socket-binding name="http" port="${jboss.http.port:8080}"/>
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="iiop" interface="unsecure" port="3528"/>
    <socket-binding name="iiop-ssl" interface="unsecure" port="3529"/>
    <socket-binding name="jgroups-mping" interface="private" port="0" multicast-address="${jboss.default.multicast.address:230.0
    <socket-binding name="jgroups-tcp" interface="public" port="7600"/>
    <socket-binding name="jgroups-tcp-fd" interface="public" port="57600"/>
    <socket-binding name="jgroups-udp" interface="private" port="55200" multicast-address="${jboss.default.multicast.address:230
    <socket-binding name="jgroups-udp-fd" interface="private" port="54200"/>
    <socket-binding name="modcluster" port="0" multicast-address="224.0.1.105" multicast-port="23364"/>
    <socket-binding name="txn-recovery-environment" port="4712"/>
    <socket-binding name="txn-status-manager" port="4713"/>
    <outbound-socket-binding name="mail-smtp">
        <remote-destination host="localhost" port="25"/>
    </outbound-socket-binding>
</socket-binding-group>
```

Ilustración 28: Configuración de jgroups-tcp con interfaces públicas. Captura de pantalla.

Una vez finalizados estos cambios se requiere reiniciar el sistema una vez más para verificar estos cambios.

```
# service wildfly restart
```

Una vez reiniciado el servidor se debería crear un archivo con la lista de servidores.

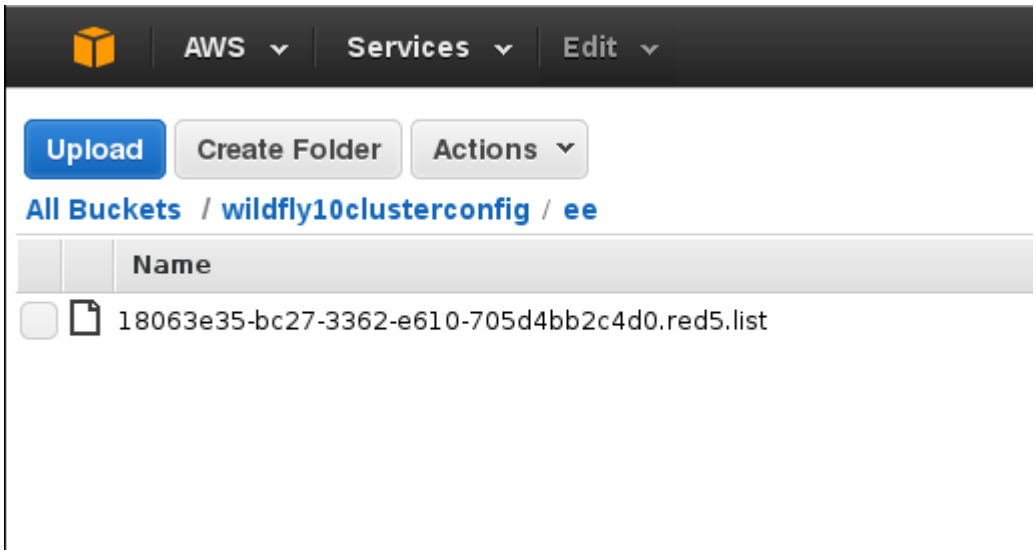

Ilustración 29: Archivo de listado de nodos del clusters. Captura de pantalla.

Este archivo está dentro de una carpeta “ee” con una lista de los servidores. Al ir aumentando los nodos se va a actualizar este archivo de texto.

```
red5    f2d8d9ff-f50b-2fad-0590-bce07a39c21c    172.30.0.133:7600
```

Ilustración 30: Ejemplo de lista de nodos.

Si alguna vez se en el listado de nodos la dirección IP de localhost 127.0.0.1, significa que no se pueden ver los servidores entre ellos. Ya sea porque no están configurados adecuadamente o porque hay problemas de red o de corta fuegos que no permiten que se vean los servidores.

Para probar la conexión entre los nodos se puede apuntar de un nodo a otro con telnet al puerto 7600.

```
$ telnet 172.30.0.133 7600
```

Siendo 172.30.133 la dirección IP interna otorgada por Amazon a nuestro servidor.

4.1.3 Creación de imágenes

Una vez creado el servidor ya podemos generar las imágenes. El tener una imagen de un servidor nos permite desplegar varios servidores con las mismas características, configuración y software.

En el panel de configuración de Amazon EC2 podemos seleccionar a nuestro servidor recientemente creado con el botón derecho del ratón, seleccionar *Image* y luego *Create Image*.

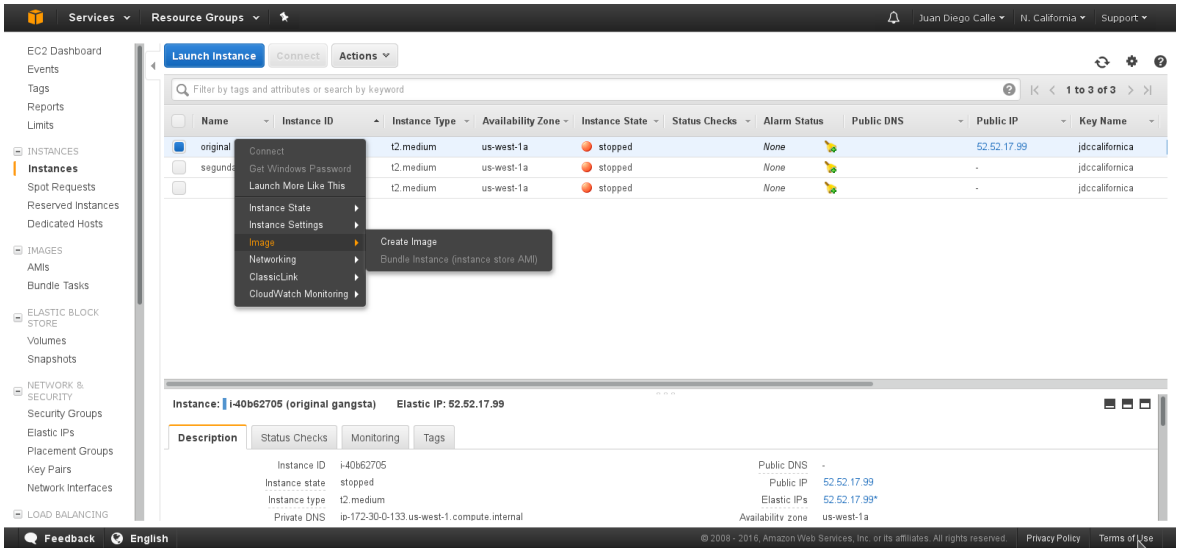

Ilustración 31: Creación de imagen a partir de un servidor EC2. Menú desplegable. Captura de pantalla.

Al momento de seleccionar *Create Image* se nos despliega una ventana con un formulario. En este formulario hace falta llenar el nombre de la imagen bajo *Image Name*. Similar a ilustración 31.

Al terminar de crear la imagen debe salir una mensaje de que ser recibió la petición para crearla. Ver ilustración 32.

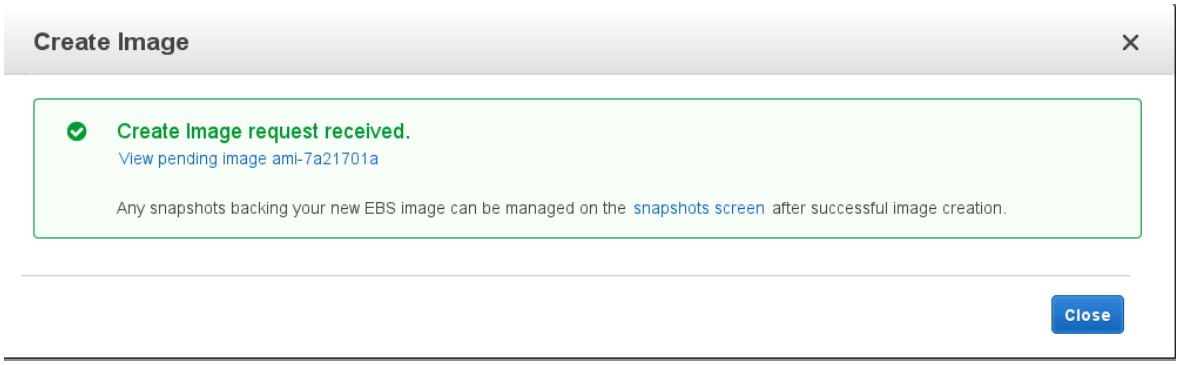

Ilustración 32: Mensaje petición de creación de imagen recibido correctamente. Captura de pantalla.

Una vez creada la imagen se la puede encontrar bajo el menú *Images* hay la opción *AMIs*. Ahora lo que falta es el balanceo de carga con auto escalación. Esto permitirá a

los servidores desplegarse a medida que haya mayor carga y disminuir la cantidad de servidores si es que esta aminora.

4.1.4 Balanceo de carga con auto-escalamiento

En el menú *Load Balancing* hay una opción *Load Balancers.* Ahí adentro se debe elegir el botón 

. Una vez adentro se visualiza el ayudante para crear balanceadores de carta. Se puede seleccionar entre *Application Load Balancer* o *Classic Load Balancer*. Para nuestro caso vamos a usar Classic Load Balancer. Ver ilustración 33.

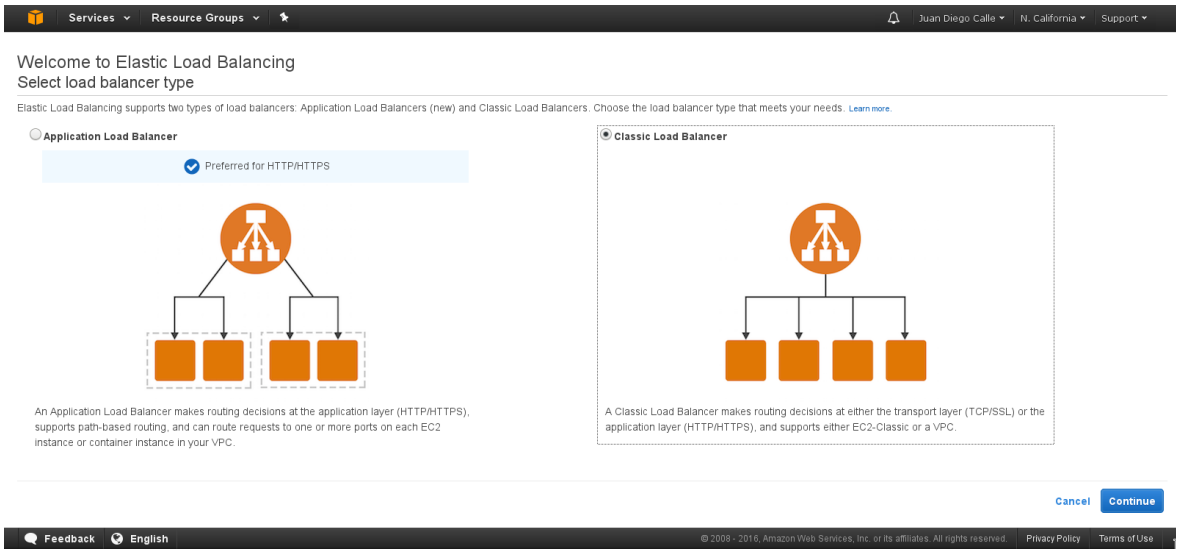

Ilustración 33: Tipos de balanceo de carga. Captura de pantalla.

Luego se debe seleccionar un nombre para el balanceador de carga y también los puertos con sus protocolos que se van a usar. Ejemplo en la ilustración 34.

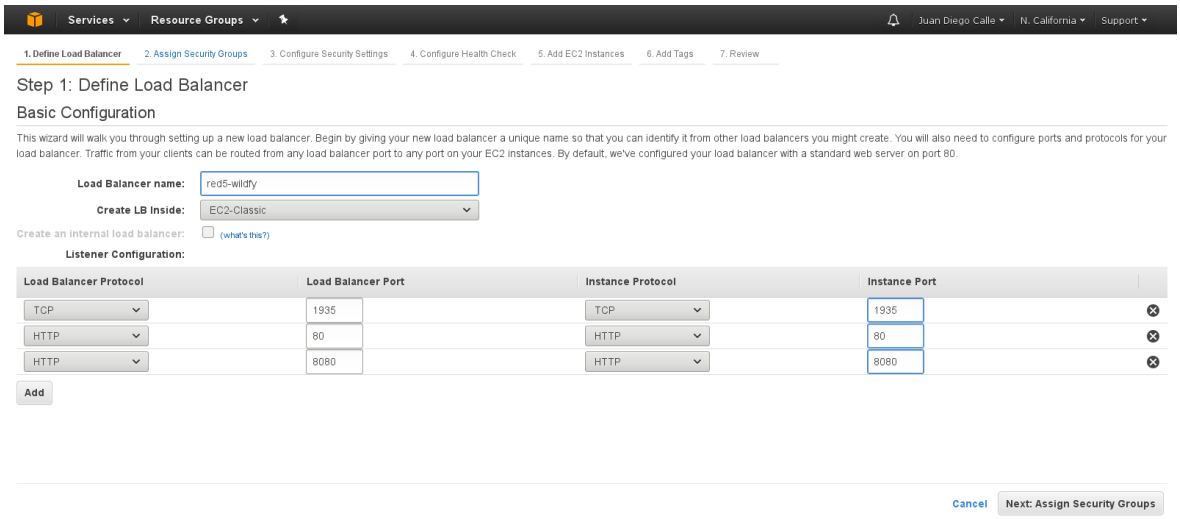

Ilustración 34: Puertos para Red5 y JBoss/Wildfly. Captura de pantalla.

En el siguiente paso es necesario agregar al balanceador de carga al grupo de seguridad que se creó antes. O se puede crear un grupo nuevo si es que se lo desea, pero es necesario que tenga acceso a los mismos puertos (1935, 80 y 8080).

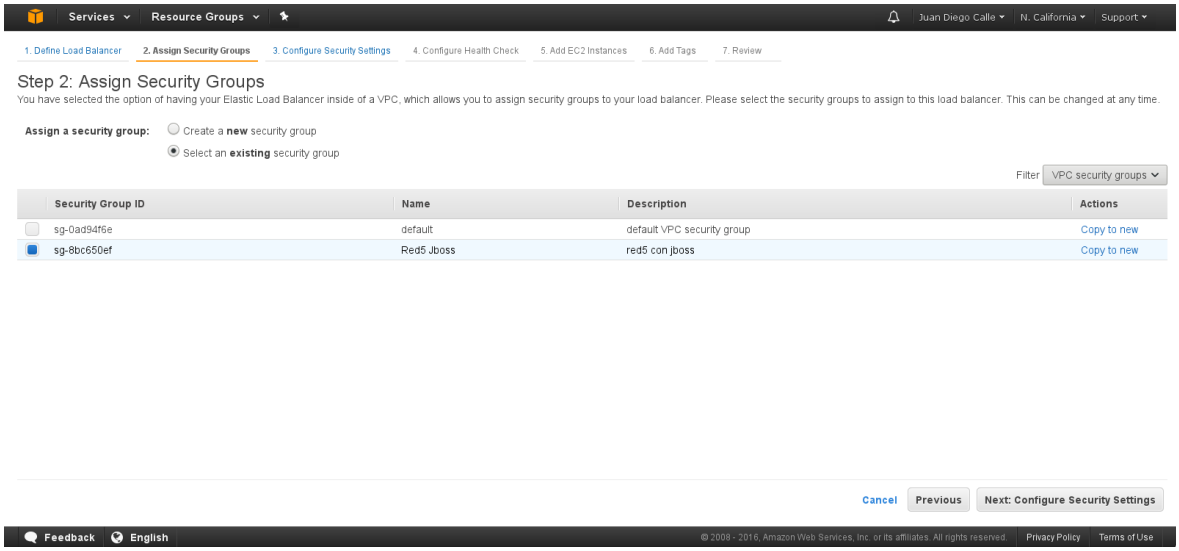

Ilustración 35: Grupos de seguridad para balanceo de carga. Captura de pantalla.

También se puede seleccionar la instancia ya creada directamente al balanceador de carga. Si es necesario se pueden agregar más instancias. Ver ilustración 36.

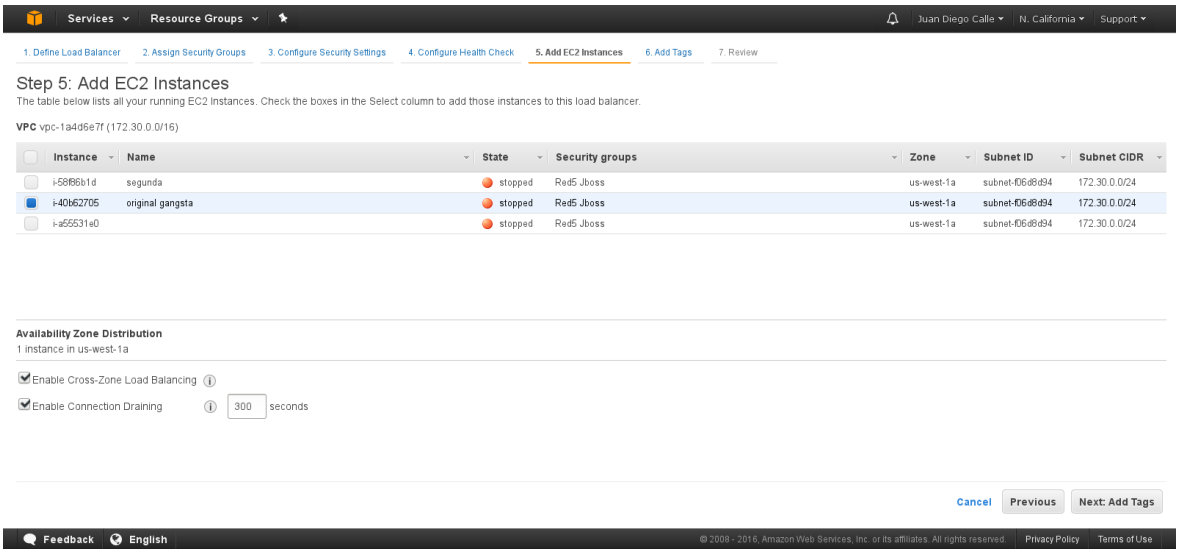

Ilustración 36: Agregar instancias directamente al balanceador de carga. Captura de pantalla.

Para proveer un mejor servicio que se ajuste a la carga de los usuarios y permita bajar costos usando solo los recursos que se necesiten se deben establecer unas políticas de auto escalamiento.

4.1.5 Grupos de auto escalamiento

Para crear un grupo de auto escalamiento es necesario ir al menú *Auto Scaling* y seleccionar *Auto Scaling Groups*. Adentro hay el botón

. Al dar clic en ese botón se debe seleccionar en el menú de la izquierda donde dice *My AMIs* donde van a estar las imágenes creadas.

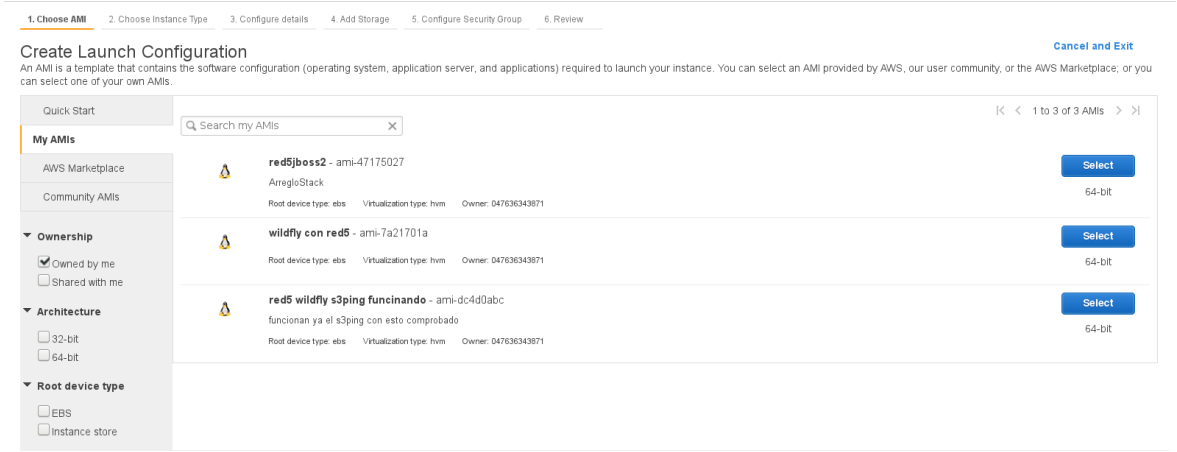

Ilustración 37: Ventana Create Launch Configuration, opción *My AMIs. Captura de pantalla.*

Y después es necesario seleccionar el tipo de instancia para usar, se debería usar por lo menos una instancia similar o con mejores requerimientos a la que usamos para crear la imagen.

Después es necesario crear unas políticas de auto escalamiento. Ver ilustración 38.

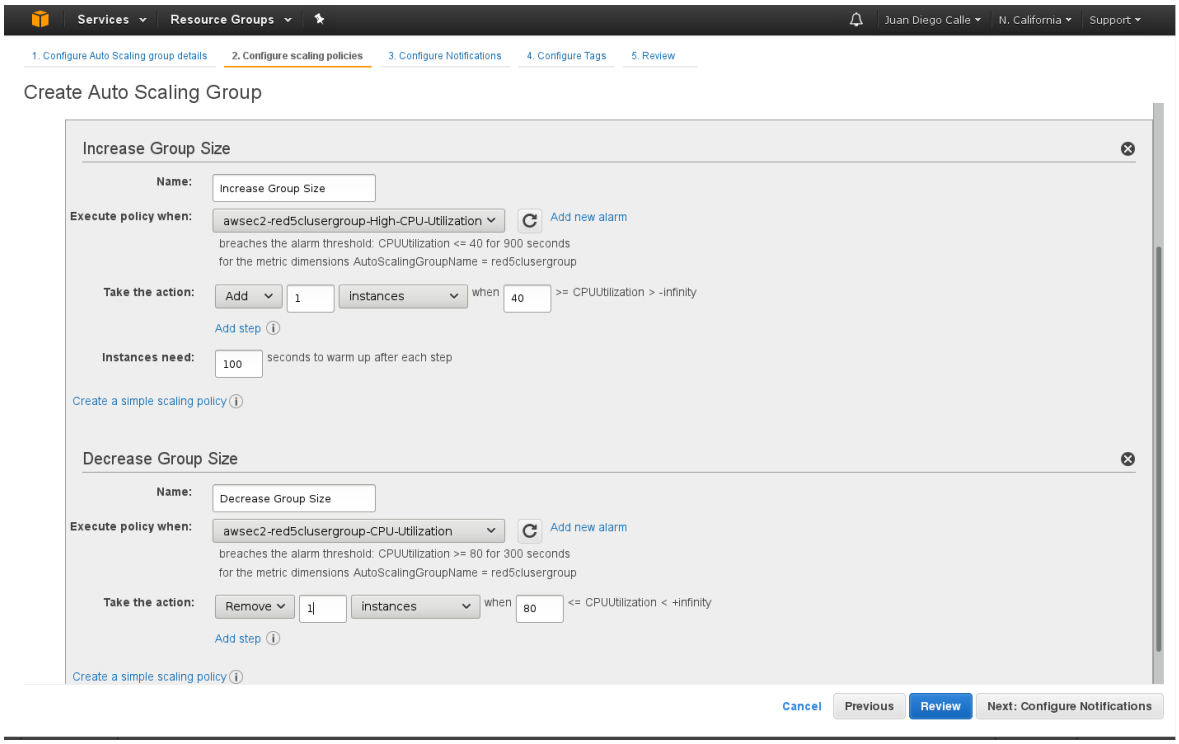

Ilustración 38: Políticas de auto escalamiento. Captura de pantalla.

Se puede elegir la saturación del CPU o RAM para el momento de crear una nueva instancia y cuánto tiempo la instancia está en desuso para apagarla.

4.2 Configurando Red5 para ser usado con JBoss

Red5 es 100% software libre por lo cual se puede descargar su código fuente del Internet. Como ya había mencionado antes el código de Red5 se lo puedo bajar de GitHub en el siguiente link https://github.com/Red5.

Para modificar a Red5 para usar JBoss/WildFly se utilizó un tutorial viejo de Red5 0.80 para JBoss AS 7 publicado en los foros de JBoss[2]. Red5 ha tenido varios cambios significativos desde la versión 0.80 hasta la versión 1.0.7 actual, de la misma forma WildFly 10.0.0 tiene 3 versiones por encima de JBoss As 7.

Para compilar Red5 se necesitan varias herramientas Java JDK 7 por lo menos para el desarrollo del proyecto, Git para poder descargar la última versión de Red5 y Maven. En base a la experiencia se va a utilizar Eclipse IDE, que tiene ya incluido Git y Maven.

2 Tomado de https://developer.jboss.org/wiki/JBossAS7Red5MediaServerAndXuggler

4.2.1 Compilación

Se necesitan 5 paquetes para compilar Red5 y además de un paquete con ejemplos de aplicaciones y clientes para este software:

- red5-parent, este es el contenedor de todos los proyectos de red5. Es el proyecto padre. Permite a todos los sub-proyectos manejar un conjunto de librerías base. Se puede descargar.(“Red5/red5-parent”, s/f)
- red5-server-common, contiene las clases y librerías en común para el servidor y los clientes.(“Red5/red5-server-common”, s/f)
- red5-service, es el servicio o programa que permite manejar a Red5. (“Red5/red5-service”, s/f)
- red5-io, librería de red5 para entrada y salida de datos. (“Red5/red5-io”, s/f)
- red5-server, es el núcleo ya la base del servidor de Red5.(“Red5/red5-server”, s/f)
- red5-examples, este quinto y último set de aplicaciones sirve para observar ejemplos de uso de clientes y aplicaciones para Red5.(“Red5/red5-examples”, s/f)

En el primer paso para compilar Red5 es descargar los repositorios de las mencionadas aplicaciones.

Para importar los proyectos nos debemos a ir a los repositorios de GitHub:

- https://github.com/Red5/red5-parent.git
- https://github.com/Red5/red5-server-common.git
- https://github.com/Red5/red5-service.git
- https://github.com/Red5/red5-io
- https://github.com/Red5/red5-server.git
- https://github.com/Red5/red5-examples.git

Desde Eclipse se requiere importar los proyectos como Git e ir al menú Archivo o File → Importar o Import. Esto se requiere hacer para todos los casos.

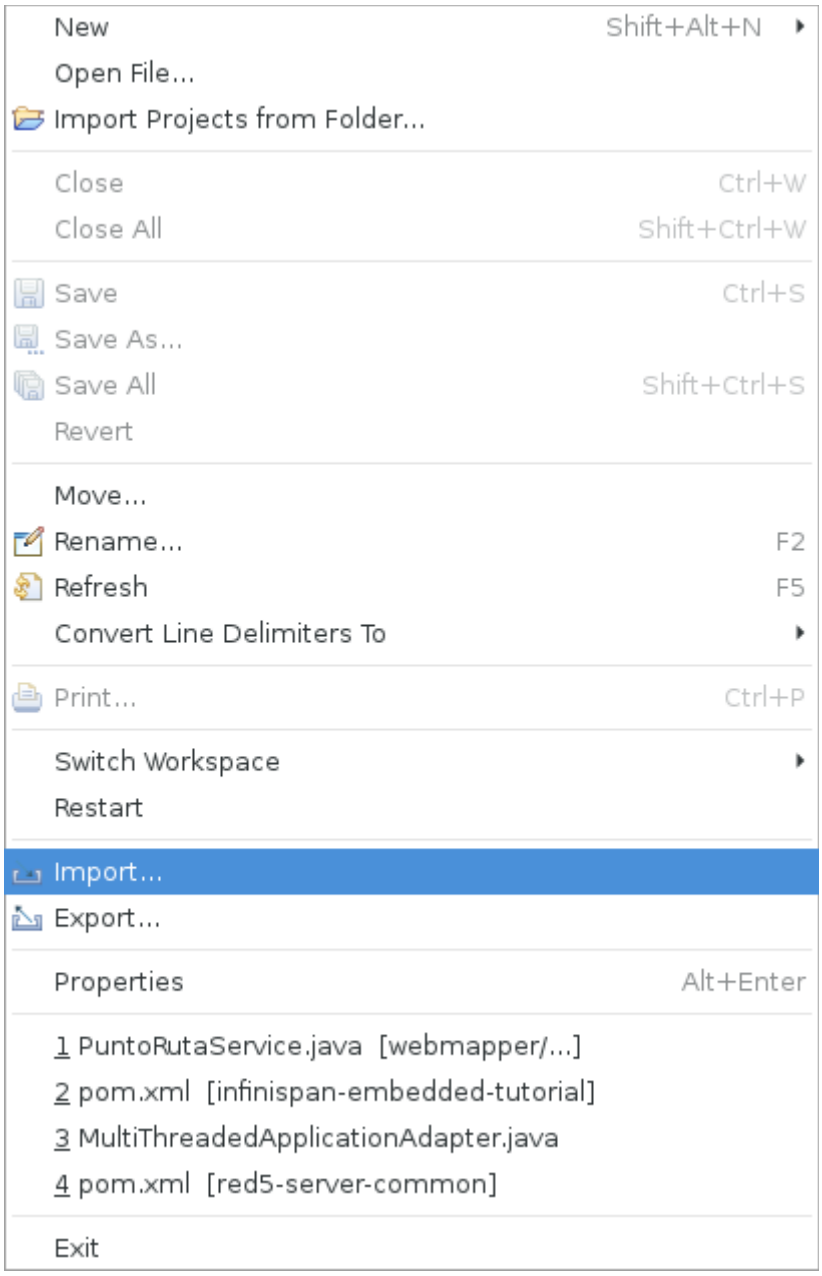

Ilustración 39: Importar desde el menú Archivo/File desde Eclipse. Captura de pantalla.

Después de seleccionar “importar” se requiere elegir el tipo de proyecto a importar, que necesariamente será un proyecto tipo Git.

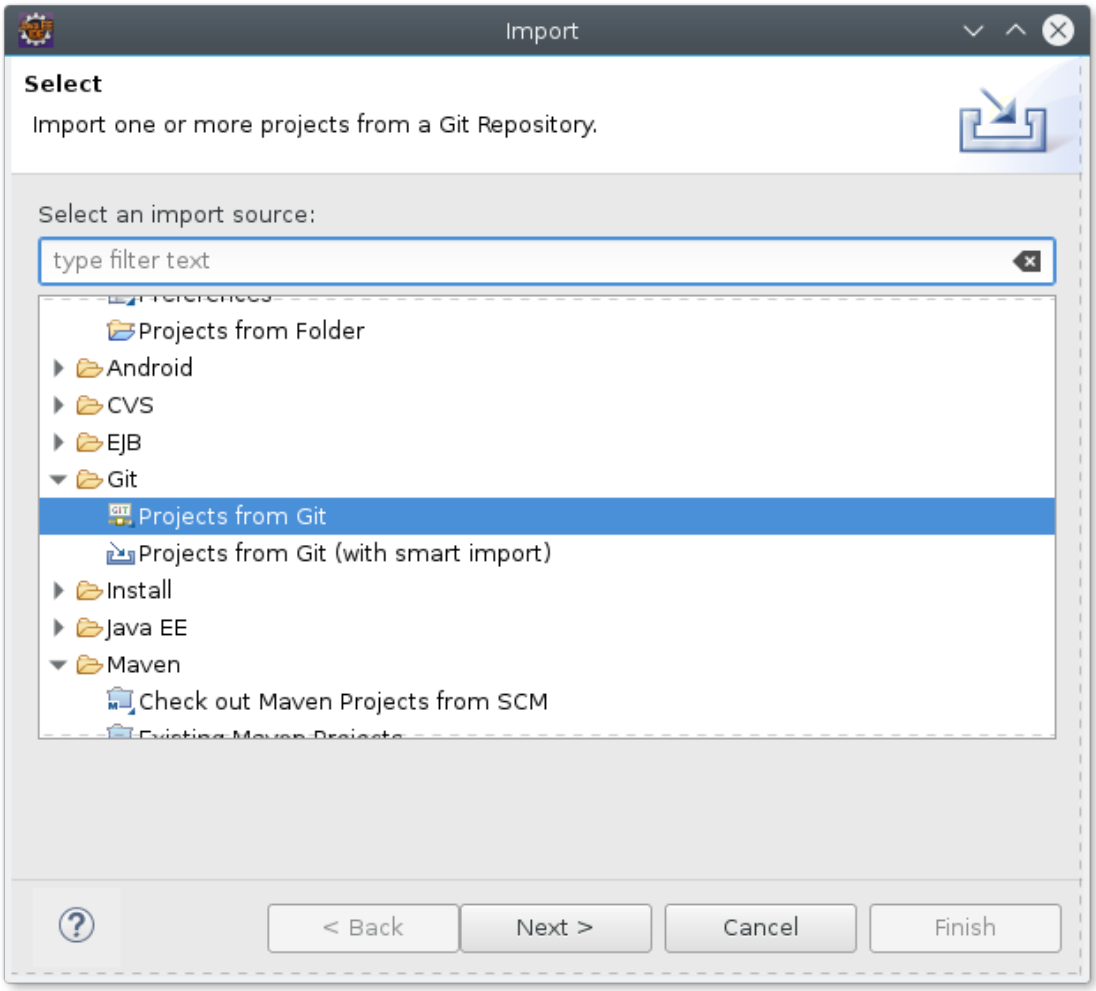

Ilustración 40: Importar proyectos desde Git. Captura de pantalla.

El siguiente paso es ir a GitHub que comprende las distintas páginas de los proyectos para obtener los repositorios a descargar.

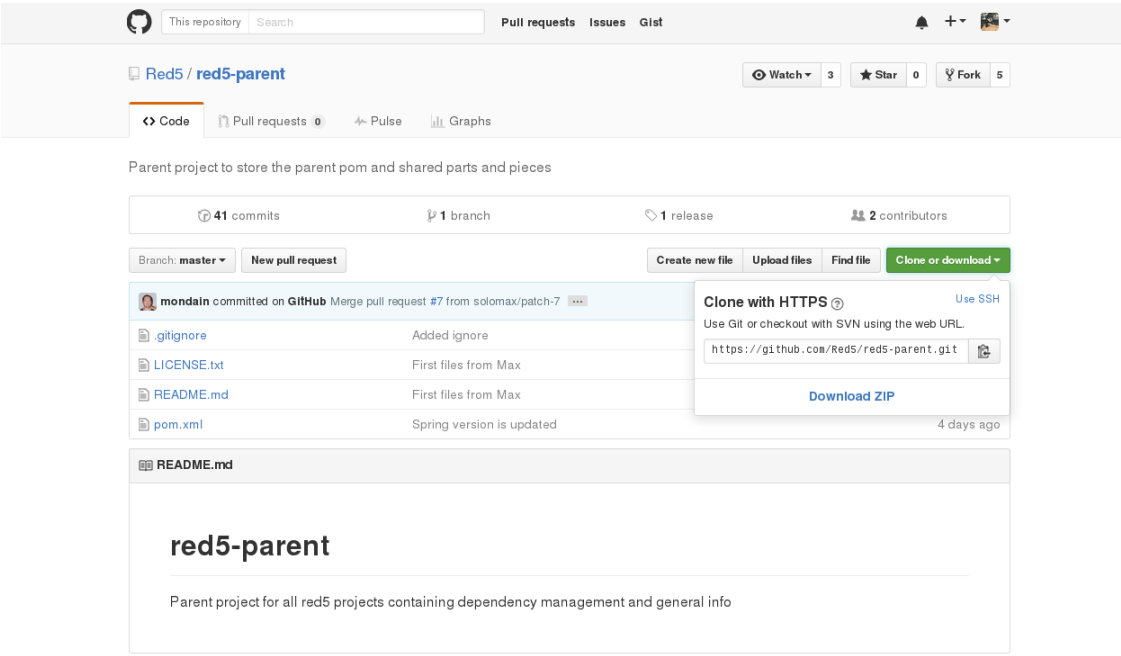

Ilustración 41: Importando proyecto desde GitHub de Red5-Parent, https://github.com/Red5/red5-parent. Captura de pantalla.

En cada proyecto de GitHub permite clonar, o *clone* en inglés, por medio de HTTPS o SSH. Cualquiera de las opciones es útil, pero para usar SSH se necesita tener un usuario registrado en GitHub con una llave también registrada. En la ilustración 33 se puede ver la dirección para clonar. Esta dirección es la que vamos a utilizar en Eclipse al momento de importar.

Ejemplo: "https://github.com/Red5/red5-parent.git"

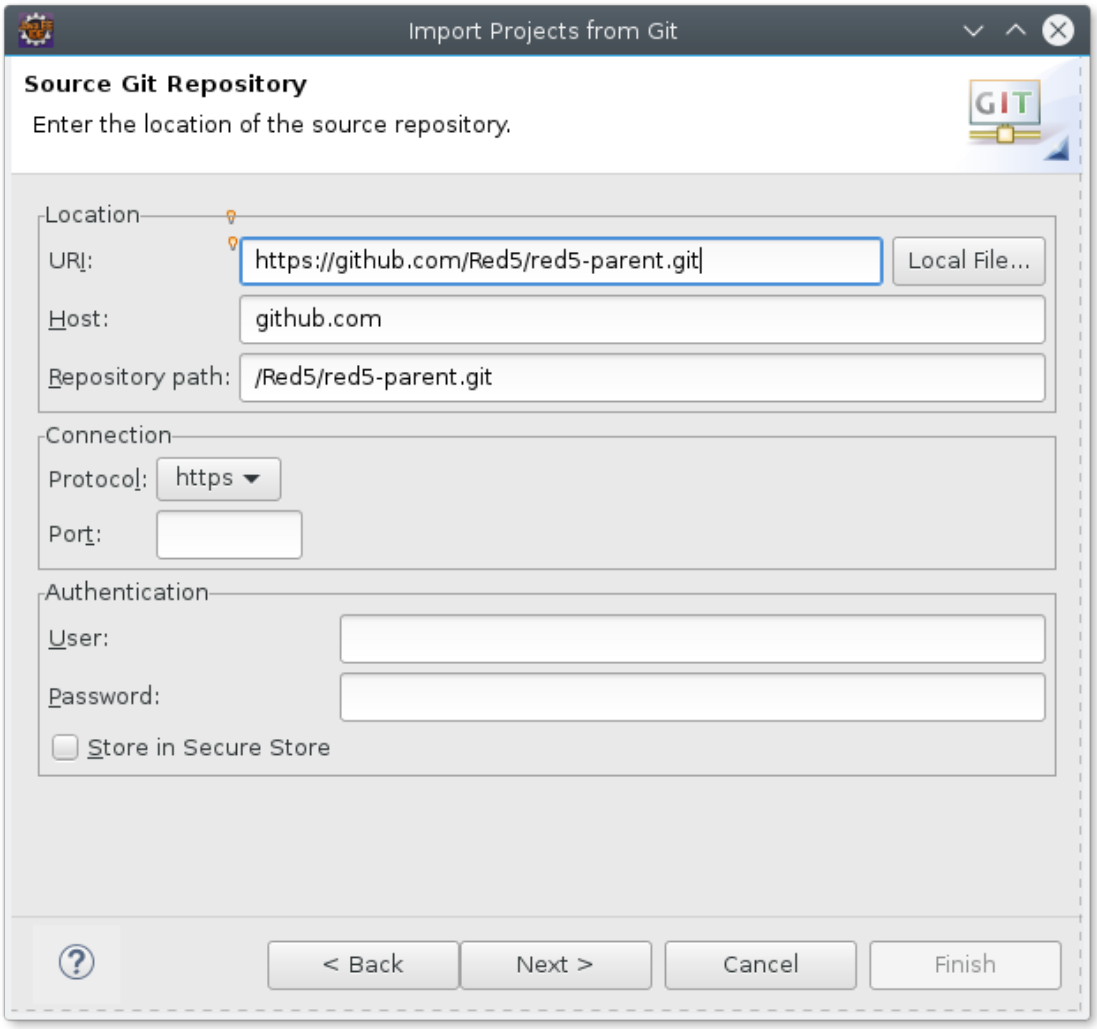

Ilustración 42: Importar repositorio de git de red5-parent.

Al copiar la dirección de git en URI se llenan automáticamente algunos campos, como *Host*, *Repository Path* y *Protocol*. Si se usa https como protocolo se debe verificar que esté como https o ssh.

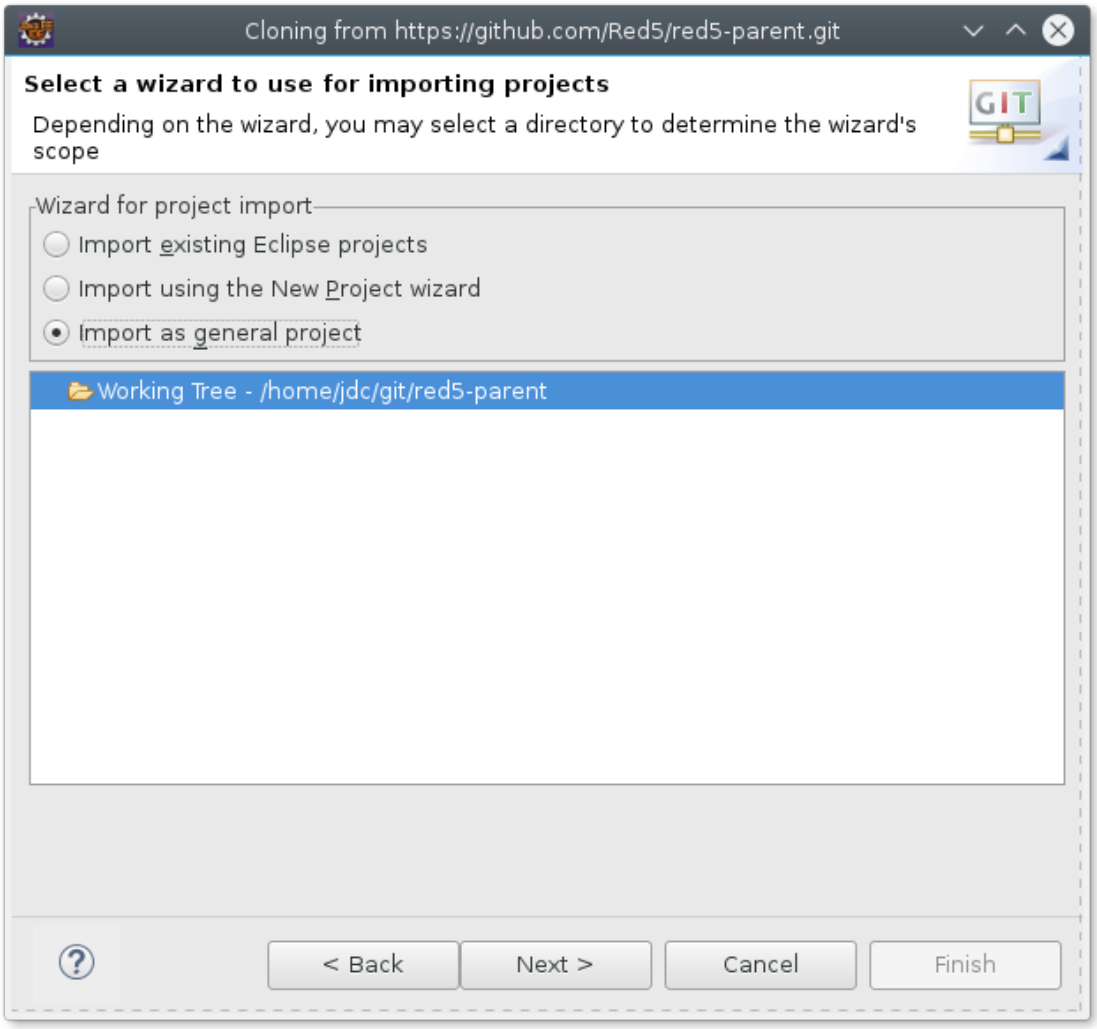

Ilustración 43: Importar como proyecto general desde Eclipse.

Lo siguiente es iar, o *clone* en inmportar como Proyecto General o *General Proyect* ya que este de por si no tiene los archivos de configuración de Eclipse para su manejo. De otra forma Eclipse no va a poder encontrar o va a tener problemas ubicando las fuentes y librerías necesarias.

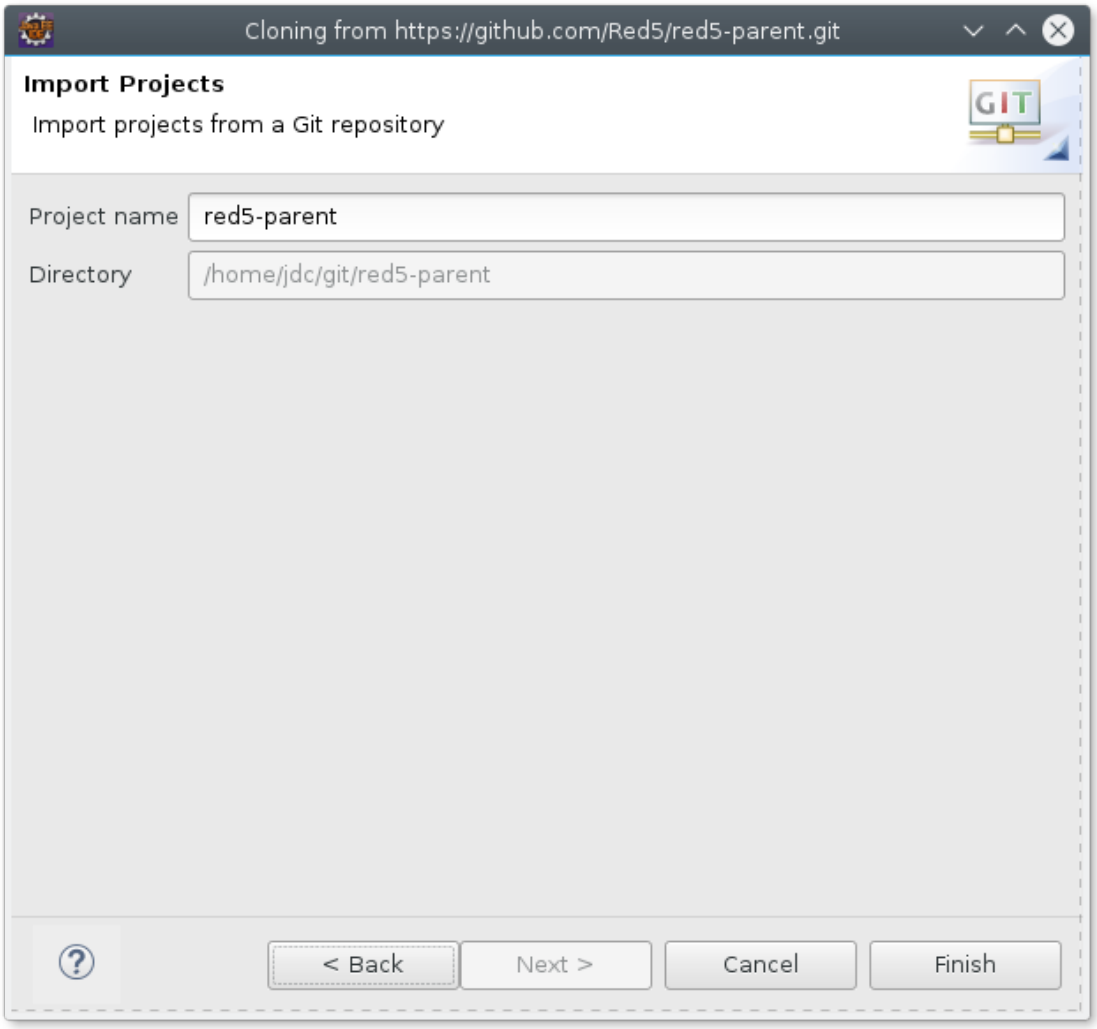

Ilustración 44: Importar proyecto de Git, última fase. Captura de pantalla.

En la última fase se puede cambiar de nombre, por motivos de simplicidad se va a mantener cada proyecto con el nombre original. El proyecto por defecto es general por lo cual es necesario convertirlo a un proyecto de Maven para que tenga soporte de las funciones necesarias en Eclipse.

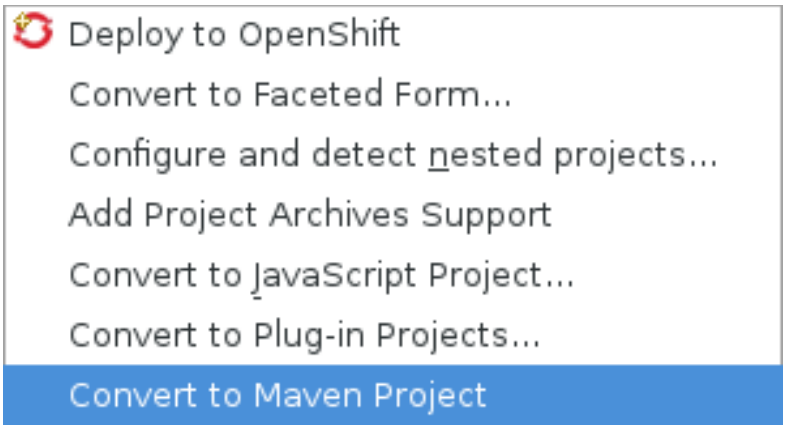

Ilustración 45: Convertir a un proyecto de Maven. Captura de pantalla.

Todos los pasos seguidos se requieren repetir para cada uno de los repositorios de Red5. Esto permitirá visualizar los proyectos de una manera más apropiada. Ver ilustración 46.

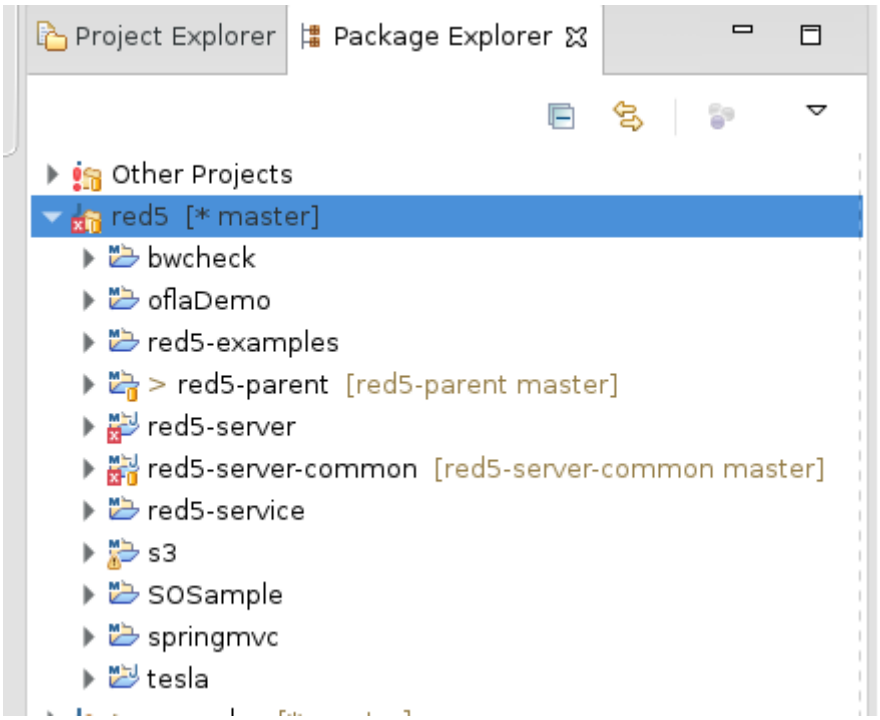

Ilustración 46: Proyectos de Red5 con ejemplos en Eclipse.

Al importar correctamente los proyectos de Red5 vamos a ver los 4 proyectos más los ejemplos que vienen en un solo proyecto. Todos estos proyectos están basados en Maven. Para construirlos solo hace falta correr el comando “maven install”.ar, o *clone* en in

4.2.2 Preparación de red5.ear

En base al tutorial de “JBoss As 7, Red5 Media Server and Xuggler”[3] vamos a construir un EAR. EAR viene del inglés **E**nterprise **Ar**chive. . Este es un formato de archivo soportado porJava EE para empaquetado de varios módulos, que tiene una extensión “ear”. Su formato real es un “zip” que contiene un directorio llamado “META-INF” que es una carpeta con los metadatos de la aplicación con los descriptores del lanzamiento o deployamiento[4] (“EAR (file format)”, 2016).

Algunos pasos no coinciden con el tutorial por lo cual se tuvo que investigar y solucionar a modo de prueba y error.

Adentro de la carpeta “standalone/deployments” en nuestro directorio donde está ubicado WildFly debemos crear una carpeta llamada “red5.ear” con 3 carpetas adentro “META-INF”, “lib” y “red5.war”.

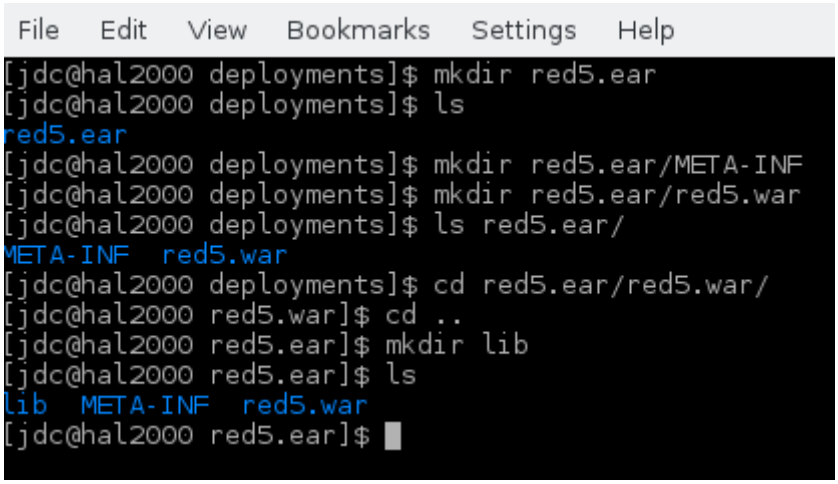

Ilustración 47: Creación de carpeta red5.ear y sus carpetas necesarias.

Después de crear las carpetas necesarias debemos ir a las fuentes de Red5-server para copiar los archivos necesarios para la parte web de este software. En el interior de las fuentes de Red5-server se tiene la carpeta “src/main/server/webapps” que varias carpetas y archivos que manejan la interfaz web Red5.

3 https://developer.jboss.org/wiki/JBossAS7Red5MediaServerAndXuggler

4 Anglicismo utilizado en el medio informático que significa despliegue de una aplicación.

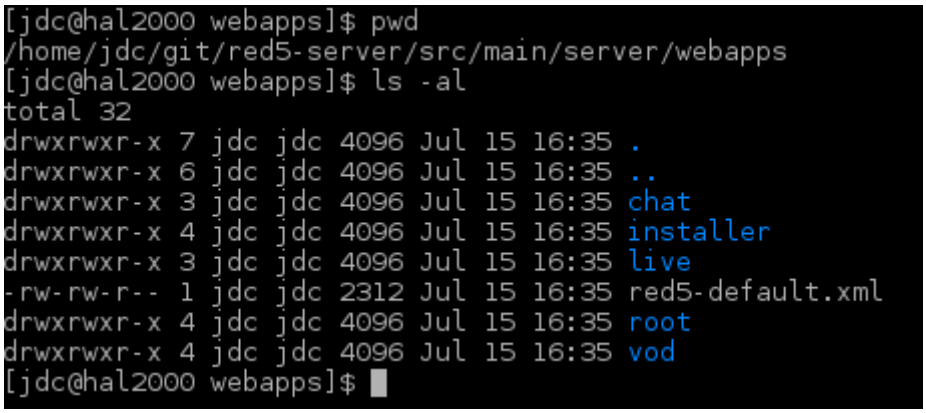

Ilustración 48: Ejemplo de la carpeta webapps de red5-server.

La carpeta *chat* tiene un ejemplo de chat, *installer* maneja los instaladores de las demostraciones, estos solo funcionan en el servidor Red5 normal, no adentro de WildFly 10. La carpeta *live* contiene un ejemplo de una aplicación en vivo, el archivo red5-default.xml contiene la configuración por defecto para levantar Red5, esta archivo no lo vamos a necesitar. Por último tenemos a la carpeta *root* y *vod*, la primera contiene la base web de Red5 que vamos a copiar a la carpeta red.war adentro de red5.ear y *vod* viene de las siglas en inglés "video on demand" con efectos de video en demanda.

Luego vamos a copiar todos los archivos de esa carpeta a la carpeta red.war que está al interior de nuestro red5.ear. El siguiente paso es modificar el archivo de configuración web.xml donde se aumentarán las siguientes líneas:

```
<context-param>
   <param-name>webAppRootKey</param-name>
   <param-value>/red5</param-value>
</context-param>
```

Lo mejor es poner estas líneas después de las siguientes.

```
<display-name>root</display-name>
```

4.2.3 Creación de red5-context.jar

El siguiente paso es crear un "jar" con los archivos de configuración y contexto para el Red5. Un jar es básica mente un archivo comprimido en formato zip, pero extensión ".jar" en vez de".zip". Se debe copiar los siguientes archivos a "beanRefContext.xml" y "defaultContext.xml", de la carpeta "src/main/server/war" y los archivos "red5-common.xml", "red5-core.xml" y "red5.xml" de la carpeta "src/main/server/conf" de las fuentes de red5-server. Ahora se puede crear la carpeta META-INF con un archivo MANIFEST.MF vacío.

La carpeta “src/main/server/war” es una carpeta que se mantiene como legado a antiguas versiones que no ha sido actualizada por lo cual los archivos “beanRefContext.xml” y “defaultContext.xml” están desactualizados. En el repositorio https://github.com/Red5/red5-server/tree/master/src/main/server se puede ver que no ha sido actualizado en 3 años.

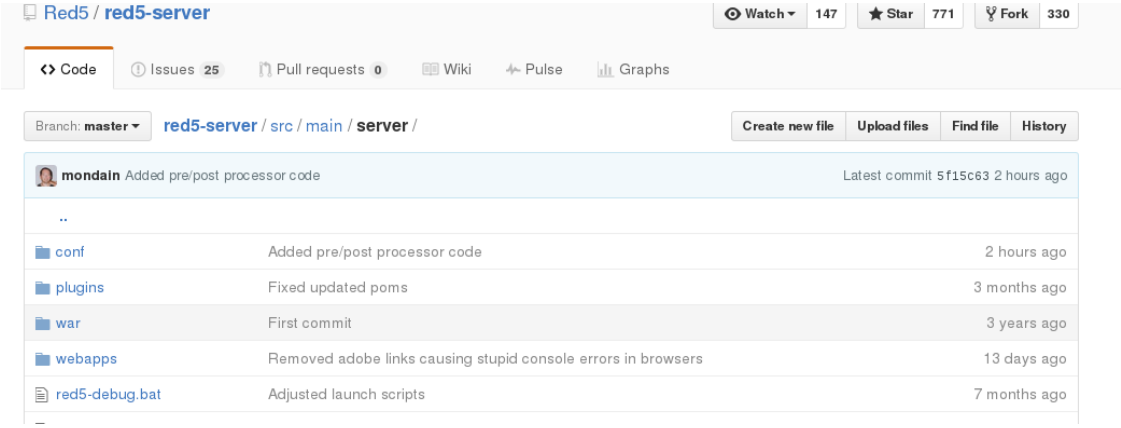

Ilustración 49: Carpeta “red5-server/src/main/server/war” en GitHub. Captura de pantalla.

El archivo “defaultContext.xml” se debe actualizar las clases porque ya no están en las carpetas y paquetes que estaban en las versiones antiguas. Entonces se debe buscar dos líneas en esos archivos. La primera es “global.scope” con la clase "org.red5.server.GlobalScope".

```
<bean id="global.scope" class="org.red5.server.GlobalScope" init-method="register" >
```

Buscando en el código fuente de “red5-server” la clase GlobalScope se encuentra en el paquete “org.red5.server.scope” por lo cual se debe cambiar la línea anterior a lo siguiente.

```
<bean id="global.scope" class="org.red5.server.scope.GlobalScope" init-method="register" >
```

También hay que buscar la clase “scopeResolver”.

```
<bean id="red5.scopeResolver" class="org.red5.server.ScopeResolver" >
```

De la misma forma que la clase anterior hay que buscar la clase “ScopeResolver” y se encontró en el paquete “org.red5.server.scope”.

4.2.4 Librerías necesarias

Maven es un software de manejo de proyectos, que permite manejar las librerías independientes del proyecto por medio de unos archivos que tienen el nombre “pom.xml”.

(“Maven – Welcome to Apache Maven”, s/f) Red5 utiliza Maven, y una de las ventajas de éste es que se pueden manejar distintos perfiles de un mismo programa. De esta forma se pueden construir varios tipos de programas al mismo tiempo con las mismas fuentes sin tener que manejar varios proyectos separados.

Si vamos al proyecto de “red5-server” y abrimos su archivo “pom.xml” abajo de la etiquetas de “profiles” o perfiles se pueden leer los distintos tipos de estos. De la misma forma en Eclipse si damos clic derecho en el proyecto y luego vamos a la opción Maven se puede ver una opción “Seleccionar Perfiles de Maven”.

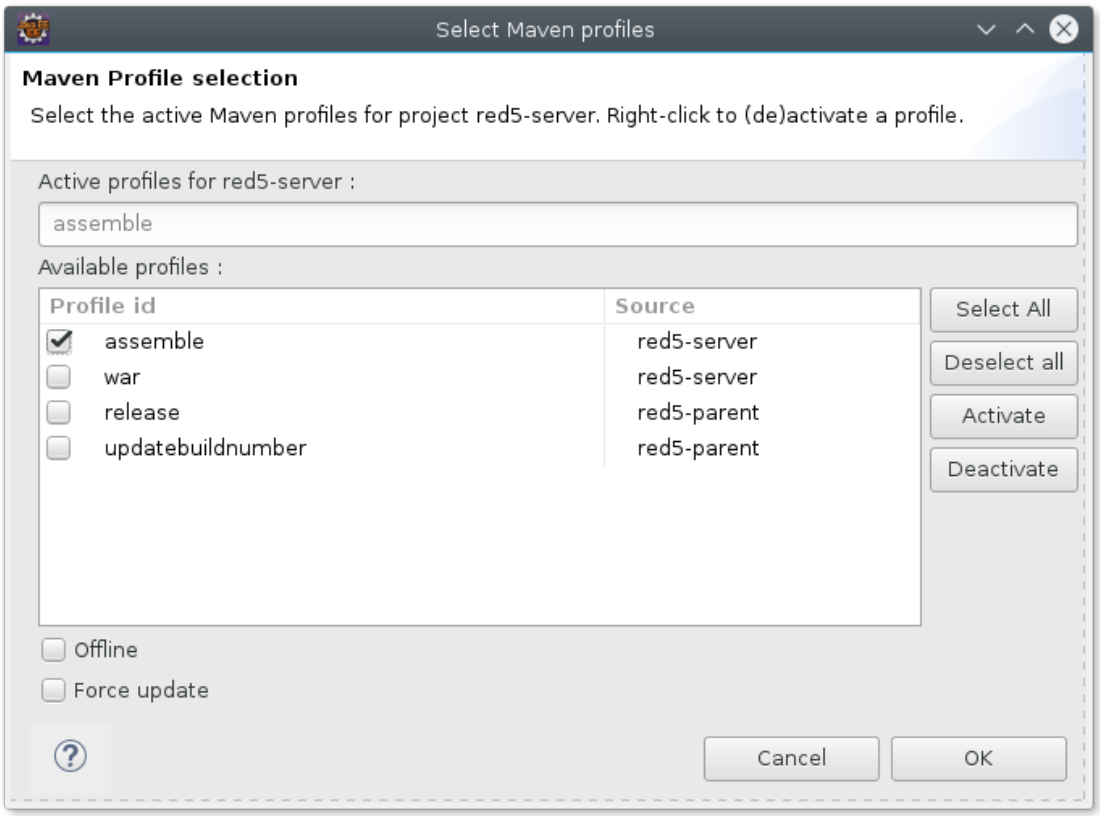

Ilustración 50: Ejemplo de Perfiles de Red5 Server. Captura de pantalla.

En el ejemplo superior se puede observar los 4 perfiles que permite “red5-server”. La opción de “war” se mantiene como legado versiones antiguas y no esta soportada. La versión “assemble” se utiliza para ensamblar el paquete de “red5-server”. Mientras que las otras 2 opciones son heredadas del proyecto padre “red5-parent”.

Si se revisa el perfil “assemble” vamos a encontrar que se puede extraer las librerías el momento de compilar el proyecto a un carpeta “lib” adentro del directorio final.

```
<outputDirectory>${project.build.directory}/lib</outputDirectory>
```

Ilustración 51: Directorio final de librerias en el perfil “assemble”.

El proyecto de “red5-server” se compila en la carpeta “target”, ahí es donde se va a encontrar la mencionada carpeta “lib”.

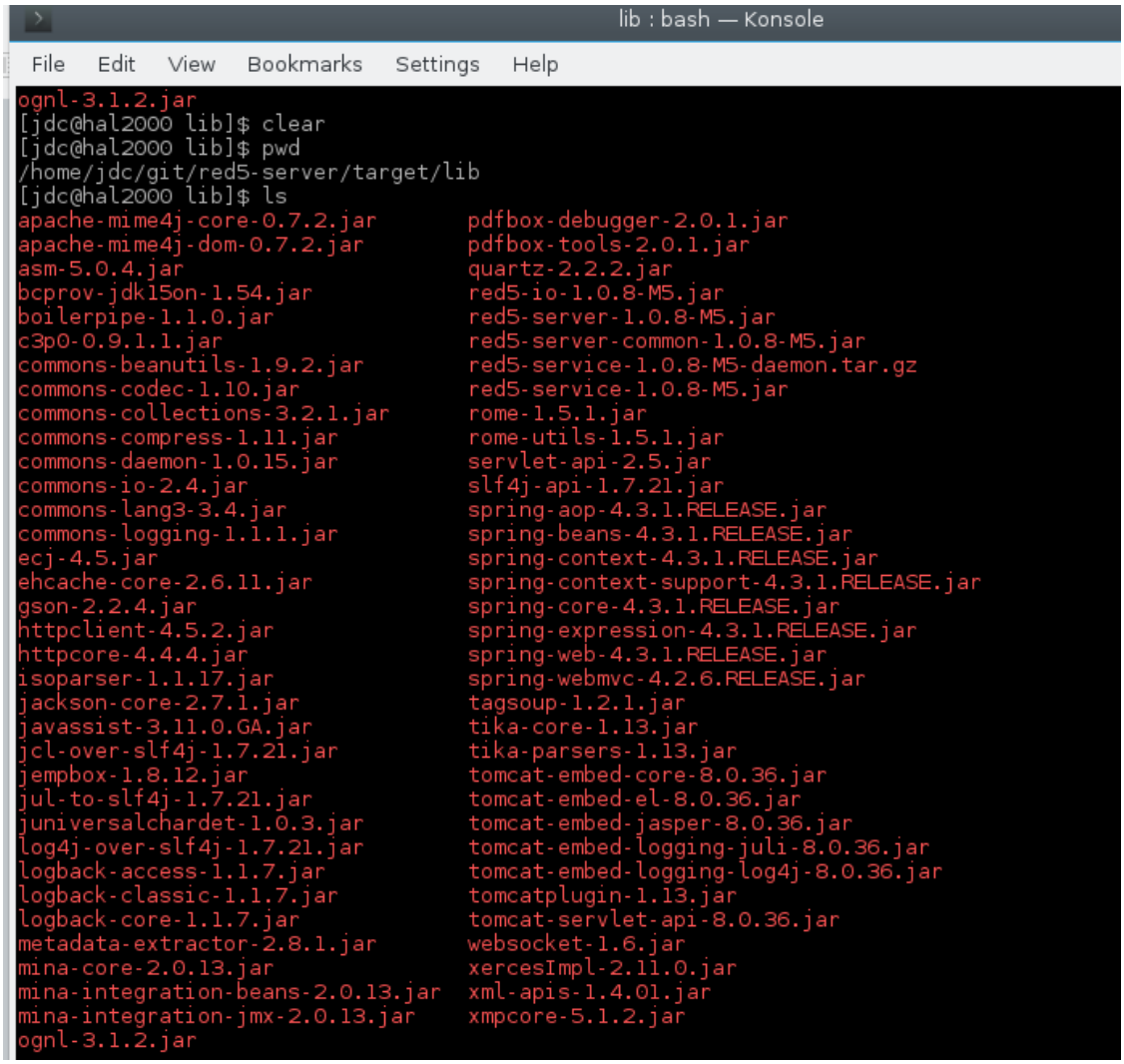

Ilustración 52: Ejemplo: librerías red5-server 1.0.8. Captura de pantalla.

Algunas de estas librerías de “red5-server” ya se encuentran adentro de JBoss/WildFly por lo cual debemos ver cuales librerías son necesarias, esto puede cambiar de versión a versión. Inicialmente en la versión de “red5-server” 1.0.6 fue necesario compilar “red5-common”, “red-service” y “red5-io” y poner los archivos “jar” en la carpeta “lib” adentro de “red5.ear”.

4.2.5 Configuración red5.ear

En la carpeta creada “META-INF” se necesitan 2 archivos: “MANIFEST.MF” y “application.xml”. En el archivo “MANIFEST.MF” solo se debe agregar la siguiente línea:

```
Dependencies: org.apache.commons.logging export, org.slf4j export
```

Ilustración 53: Dependencias necesarias en el archivo MANIFEST.MF

El archivo “application.xml” sirve para indicar las aplicaciones que van a correr adentro de nuestro EAR. Para probar Red5 se necesita una aplicación, Red5 provee varios ejemplos para este caso solo vamos a instalar 2 aplicaciones: VOD que viene del inglés “video on demand y oflademo”, que permite manejar distintos tipos de conexiones para varios ejemplos.

```
<?xml version="1.0" encoding="UTF-8"?>

<application xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/application_5.xsd"

version="5">

   <display-name>red5</display-name>

   <module>

      <web>

         <web-uri>red5.war</web-uri>

         <context-root>/red5</context-root>

      </web>

   </module>

   <module>

      <web>

         <web-uri>vod.war</web-uri>

         <context-root>/red5/vod</context-root>

      </web>

   </module>

   <module>

      <web>

         <web-uri>oflaDemo.war</web-uri>

         <context-root>/oflaDemo</context-root>

      </web>

   </module></application>
```

Ilustración 54: Ejemplo de archivo “application.xml”.

En el ejemplo anterior se puede observar las dos aplicaciones VOD y OflaDemo. Si se requiere instalar otra aplicación en esta sección se debe aumentar como módulo a la aplicación.

4.2.6 Instalación de aplicaciones (Red5, VOD y OflaDemo)

Para instalar el módulo de Red5 debemos copiar el archivo “web.xml” de la carpeta “src/main/server/webapps/root/WEB-INF” de las fuentes de “red5-server” en el caso de tenerlo antes. Después se debe agregar las siguientes líneas:

```
<context-param>
        <param-name>webAppRootKey</param-name>
        <param-value>/red5</param-value>
</context-param>
```

Ilustración 55: Líneas para agregar en “web.xml” de “red5.war”.

Es ahí cuando se debe crear una carpeta llamada “vod.war” al interior de “red5.ear”. Luego se requiere copiar el contenido de la carpeta “src/main/server/webapps/vod” de “red5-server” a la carpeta “vod.war”. Por último crear una carpeta llamada “classes” en la carpeta “WEB-INF”.

El siguiente paso es mover el archivo “WEB-INF/red5-web.xml” que se encuentra en “vod.war” hacia la carpeta “classes” re-nombrarlo a “vod-web.xml”. Luego es necesario modificar este archivo para que no llame a “red5-web.properties”.

En el archivo “web.xml” de “vod.war” se debe modificar al archivo de acuerdo a “red5-web.properties”.

```
<?xml version="1.0" encoding="UTF-8" ?>

<beans xmlns="http://www.springframework.org/schema/beans"

   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

   xmlns:lang="http://www.springframework.org/schema/lang"

                              xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd

                                                    http://www.springframework.org/schema/lang
http://www.springframework.org/schema/lang/spring-lang.xsd">

      <bean id="web.context" class="org.red5.server.Context" autowire="byType" />

      <bean id="web.scope" class="org.red5.server.scope.WebScope">

            <property name="server" ref="red5.server" />

            <property name="parent" ref="global.scope" />

            <property name="context" ref="web.context" />

            <property name="handler" ref="web.handler" />

            <property name="contextPath" value="/vod" />

            <property name="virtualHosts" value="*" />

      </bean>

      <bean id="web.handler" class="org.red5.server.adapter.ApplicationAdapter" />

</beans>
```

Ilustración 56: Ejemplo de archivo final de “vod-web.xml”.

En el archivo “web.xml” se debe aumentar las siguientes líneas:

```
<context-param>
    <param-name>globalScope</param-name>
    <param-value>default</param-value>
</context-param>
<context-param>
    <param-name>parentContextKey</param-name>
    <param-value>default.context</param-value>
</context-param>
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>WEB-INF/classes/*-web.xml</param-value>
</context-param>
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

Ilustración 57: Líneas para agregar a "web.xml" de "vod.war".

Después de debe cambiar webAppRootKey del "web.xml" de "vod.war" a "/red5/vod".

```
<context-param>
    <param-name>webAppRootKey</param-name>
    <param-value>/red5/vod</param-value>
</context-param>
```

Ilustración 58: Ejemplo: de webAppRootKey de "web.xml" de "vod.war".

De esta forma puede correr la interfaz web de la aplicación de VOD bajo el entorno de "/red5/vod". Subsecuente, al interior de "/red5.ear/red.war/WEB-INF/" es necesario crear una carpeta "classes" similar a lo que hicimos en "vod.war" y copiar 2 archivos aquí. El primero "root-web.xml" del repositorio de git de "red5-server" que se encuentra en la carpeta red5-server" "/src/main/server/war/ ". El segundo paso es mover el archivo "red5-web.xml" de la carpeta "red5.ear/red5.war/WEB-INF" y renombrarlo "installer-web.xml".

Para que los programas funcionen se debe cambiar la propiedad contextPath de “root-web.xml” de “@context.path@” a “/”. Esto se lo hace por simplicidad, la otra forma es utilizar los archivos de propiedades para modificar esta variable de contexto.

```
<property name="contextPath" value="/" />

<property
```

Ilustración 59: Ejemplo de contextPath de “root-web.xml”.

De esta forma puede correr la interfaz web de la aplicación de VOD bajo el entorno de “/red5/vod”. Subsecuente, al interior de “/red5.ear/red.war/WEB-INF/” es necesario crear una carpeta “classes” similar a lo que hicimos en “vod.war” y copiar 2 archivos aquí. El primero “root-web.xml” del repositorio de git de “red5-server” que se encuentra en la carpeta red5-server” “/src/main/server/war/ ”. El segundo paso es mover el archivo “red5-web.xml” de la carpeta “red5.ear/red5.war/WEB-INF” y renombrarlo “installer-web.xml”.

Para que los programas funcionen se debe cambiar la propiedad contextPath de “root-web.xml” de “@context.path@” a “/”. Esto se lo hace por simplicidad, la otra forma es utilizar los archivos de propiedades para modificar esta variable de contexto.

```
<property name="contextPath" value="/" />

<property
```


Ilustración 60: Ejemplo de contextPath de “root-web.xml”.

En el archivo “installer-web.xml” vamos a remover la línea de “/WEB-INF/red5-web.properties” ya que no usamos las variables de este archivo de propiedades similar a lo que se hizo con “vod-web.xml”.

```
<bean id="placeholderConfig"
class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="location" value="/WEB-INF/red5-web.properties" />

</bean>
```

Ilustración 61: Líneas a remover de “installer-web.xml”.

Es indispensable agregar y cambiar algunas líneas en base a re5-web.properties. Para ello hay que ver las variables contextPath y virtualhost y aumentar “installer service” como en el siguiente ejemplo:

```
<?xml version="1.0" encoding="UTF-8" ?>

<beans xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xmlns:lang="http://www.springframework.org/schema/lang"

    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.1.xsd

    http://www.springframework.org/schema/lang
http://www.springframework.org/schema/lang/spring-lang-3.1.xsd">

        <bean id="web.context" class="org.red5.server.Context" autowire="byType" />

        <bean id="web.scope" class="org.red5.server.scope.WebScope" init-
method="register">

                <property name="server" ref="red5.server" />

                <property name="parent" ref="global.scope" />

                <property name="context" ref="web.context" />

                <property name="handler" ref="global.handler" />

                <property name="contextPath" value="/installer" />

                <property name="virtualHosts" value="*" />

        </bean>

        <bean id="web.handler" class="org.red5.server.adapter.ApplicationAdapter" />

        <bean id="installer.service" class="org.red5.server.service.Installer">

                <property name="applicationRepositoryUrl" value="http://red5.org/snapshots/" />

        </bean>

</beans>
```

Ilustración 62: Ejemplo final de "installer-web.xml".

4.3 Problemas de arranque Wildfly

Si el sistema no arranca puede darse por conflicto de varios paquetes de Java. Si está instalando Java OpenJDK 7 y Java OpenJDK 8 al mismo tiempo. Si este es el caso seria mejor desinstalar Java OpenJDK 7.

```
# yum remove java-1.7.0-openjdk-1.7.0.101-2.6.6.1.67.amzn1
```

Error de UnknownHostException.

```
Caused by: java.net.UnknownHostException: ip-172-30-0-9: ip-172-30-0-9: unknown error
        at java.net.InetAddress.getLocalHost(InetAddress.java:1505)
        at com.sun.corba.se.impl.orb.ORBImpl.getLocalHostName(ORBImpl.java:1740)
        ... 8 more
Caused by: java.net.UnknownHostException: ip-172-30-0-9: unknown error
        at java.net.Inet4AddressImpl.lookupAllHostAddr(Native Method)
        at java.net.InetAddress$2.lookupAllHostAddr(InetAddress.java:928)
        at java.net.InetAddress.getAddressesFromNameService(InetAddress.java:1323)
        at java.net.InetAddress.getLocalHost(InetAddress.java:1500)
        ... 9 more
```

Ilustración 63: Error de Unknown host en WildFly.

El error causado por UnknowHostException se da debido a que JBoss/WildFly necesita que se utilice un hostname o nombre de servidor, y en los Amazon Linux AMIs por defecto utilizan el DNS de Amazon y por defecto utilizan localhost. Por lo cual debemos asignar un nombre al servidor.

Lo primero que debemos hacer es modificar el archivo “/etc/sysconfig/network”, cambiar la variable HOSTNAME. Para nuestro caso voy a llamar red5.localdomain.

```
HOSTNAME=red5.localdomain
```

Luego debemos modificar el archivo /etc/hosts para que ese nombre se pueda resolver.

```
127.0.0.1   red5.localdomain red5 localhost localhost.localdomain
```

Debido a que los archivos de configuración pueden variar entre versiones o en algunos casos las clases pueden cambiar de lugar es necesario rastrear las clases. En algunos casos JBoss/Wildfly puede tener algunas de estas librerías por defecto por lo cual se deben rastrear las librerías manualmente y tener un conocimiento de Java y Maven para poder solucionarlos.

Al tener a Red5 corriendo sin problemas se debe crear una aplicación para que este mismo permita la comunicación y replicación de información entre nodos retransmitiendo los streams.

Capítulo 5: Aplicación para Red5 y análisis comparativo para Red5

Para que funcione en totalidad el cluster y finalizar con el cumplimiento de la aplicación de software libre para streaming de alta disponibilidad. Es necesario crear una aplicación de Red5 que permita al resto de nodos mantener la información que se está transmitiendo por un nodo. Se puede usar la base de los ejemplos otorgados por Red5 en el repositorio de GitHub https://github.com/Red5/red5-examples.

La base que se usó para el proyecto fue la de OflaDemo y se puede limpiar todo lo que no se utiliza de OflaDemo. Por lo tanto es preferible borrar las carpetas “groovy”, “javascript”, “python”, “resources” y“ruby” de la carpeta “usr/main”. Es importante mantener la carpeta “java” y “webapp”.

El siguiente paso es aumentar algunas dependencias al archivo pom.xml:

```
<dependency>
        <groupId>org.red5</groupId>
        <artifactId>red5-client</artifactId>
        <version>${red5-client.version}</version>
 </dependency>
 <dependency>
        <groupId>org.wildfly</groupId>
        <artifactId>wildfly-server</artifactId>
        <version>8.2.1.Final</version>
 </dependency>
 <dependency>
        <groupId>org.infinispan</groupId>
        <artifactId>infinispan-embedded</artifactId>
        <version>8.2.1.Final</version>
 </dependency>
 <dependency>
        <groupId>org.jgroups</groupId>
        <artifactId>jgroups</artifactId>
        <version>3.6.10.Final</version>
 </dependency>
```

Ilustración 64: Dependencias agregadas al pom.xml de OflaDemo.

Se puede modificar el pom.xml para cambiar el “artifactId” o nombre generado de nuestra aplicación. Esto es recomendable debido a que ya se está usando OflaDemo como nombre en algunos de los ejemplos y de esta manera se evitan conflictos.

Para que pueda usar estas librerías es necesario agregarlas a la carpeta “lib” adentro de red5.ear. En este caso se va agregar la librería red5-client ya que wildfly-server, infinispan-embedded y jgroups están contenidas por defecto adentro de Wildfy.

Para concluir con la aplicación además es necesario un análisis comparativo de costos y opiniones.

5.1 Jboss-deployment-structure.xml

Al usar crear un EAR es necesario llamar a las librerías que la aplicación interna de Red5 va a usar. La forma adecuada es por medio de un archivo "jboss-deployment-structure.xml" o por el archivo "MANIFEST.MF" adentro de la carpeta "META-INF".

```
<jboss-deployment-structure xmlns="urn:jboss:deployment-structure:1.2">
 <sub-deployment name="tesla-1.0.war">
  <dependencies>
    <module name="org.infinispan" slot="main" />
    <module name="org.jboss.as.server" slot="main" />
    <module name="org.jgroups" slot="main"/>
  </dependencies>
 </sub-deployment>
</jboss-deployment-structure>
```

Ilustración 65: Ejemplo del archivo jboss-deployment-structure.xml

5.2 Application.java

La aplicación de Red5 tiene una clase principal donde se maneja la base de la aplicación. Este archiv

o se llama "Aplication.java" y debe extender la clase MultiThreadedApplicationAdapter o ApplicationAdapter que son clases creadas por Red5 para usar en las aplicaciones.

Para este trabajo se utilizó OflaDemo y https://github.com/Red5/red5-rtsp-restreamer/blob/master/src/main/java/org/red5/demos/rtsp/Application.java un ejemplo de Red5 para hacer Restreaming ("Red5/red5-rtsp-restreamer", s/f).

Se va a usar Jgroups e Infinispan para mandar la información hacia los otros nodos. La siguiente función se agregó para detectar los otros nodos.

```
public Collection<IpAddress> getClusterMembersJGroups() {
	Channel channel = (Channel) CurrentServiceContainer.getServiceContainer()
			.getService(ServiceName.JBOSS.append("jgroups", "channel",
"web")).getValue();
	List<org.jgroups.Address> members = channel.getView().getMembers();
	List<IpAddress> addresses = new ArrayList<IpAddress>();
	for (org.jgroups.Address member : members) {
		PhysicalAddress physicalAddr = (PhysicalAddress) channel
				.down(new Event(Event.GET_PHYSICAL_ADDRESS,
member));
			IpAddress ipAddr = (IpAddress) physicalAddr;
			addresses.add(ipAddr);
	}
	return addresses;
}
```

Ilustración 66: Función para obtener los nodos de JGroups.

Una vez que se tiene la función podemos retransmitir al resto de nodos.

Para hacer un Restreaming se debe usar la siguiente función:

```
public void streamBroadcastStart(IBroadcastStream stream)
{
    IScope scope = stream.getScope();
    IBroadcastScope bsScope = getBroadcastScope(scope, stream.getPublishedName());
    StreamingProxy proxy = new StreamingProxy();
    proxy.setHost(myhost);
    proxy.setApp("app");
    proxy.setPort(1935);
    proxy.init();
    bsScope.subscribe(proxy, null);
    proxy.start("MY_STRING", StreamingProxy.LIVE, null);
    streamingProxyMap.put(stream.getPublishedName(), proxy);
    stream.addStreamListener(this);
}
```

Ilustración 67: Función de ejemplo para hacer Restreaming

En la función superior es un ejemplo para manejar restreaming. Se puede ajustar para una lista de nodos.

El ejemplo de código se va a subir a GitHub. https://github.com/jdc18/red5-jboss-cluster.

5.3 Análisis Comparativo

Los aspectos tomados en consideración para este análisis son: los costos y los comentarios de los usuarios de tal manera de poner en relación nuestra propuesta y demostrar que el streaming de alta disponibilidad con software libre usando JBoss y Red5 es una solución viable.

5.3.1 Costos

Los costos en Amazon Web Services pueden variar de acuerdo a la zona y no todos los tipos de instancias sirven para todo.

Tipo de Instancia	Software	EC2	Total
m1.large	$0.23/hr	$0.175/hr	$0.405/hr
m1.xlarge	$0.97/hr	$0.35/hr	$1.32/hr
m2.xlarge	$0.23/hr	$0.245/hr	$0.475/hr
m2.2xlarge	$3.76/hr	$0.49/hr	$4.25/hr
m2.4xlarge	$3.76/hr	$0.98/hr	$4.74/hr
m3.large	$0.23/hr	$0.133/hr	$0.363/hr
m3.xlarge	$0.46/hr	$0.266/hr	$0.726/hr
c1.xlarge	$0.89/hr	$0.52/hr	$1.41/hr

Ilustración 68: Costos de Adobe Media Server en Amazon en US East Virginia. Fuente: "AWS Marketplace: Adobe Media Server 5 Extended", s/f

Costo de Wowza

Tipo de Instancia	Software	EC2	Total
t2.small	$0.157/hr	$0.026/hr	$0.183/hr
t2.medium	$0.147/hr	$0.052/hr	$0.199/hr
m3.medium	$0.188/hr	$0.067/hr	$0.255/hr
m3.large	$0.276/hr	$0.133/hr	$0.409/hr
m3.xlarge	$0.488/hr	$0.266/hr	$0.754/hr
m3.2xlarge	$0.546/hr	$0.532/hr	$1.078/hr
c3.large	$0.181/hr	$0.105/hr	$0.286/hr
c3.xlarge	$0.201/hr	$0.21/hr	$0.411/hr
c3.2xlarge	$0.401/hr	$0.42/hr	$0.821/hr
c3.4xlarge	$0.531/hr	$0.84/hr	$1.371/hr
c3.8xlarge	$0.661/hr	$1.68/hr	$2.341/hr
r3.large	$0.29/hr	$0.166/hr	$0.456/hr
r3.xlarge	$0.459/hr	$0.333/hr	$0.792/hr
r3.2xlarge	$0.661/hr	$0.665/hr	$1.326/hr
r3.4xlarge	$0.79/hr	$1.33/hr	$2.12/hr
r3.8xlarge	$1.09/hr	$2.66/hr	$3.75/hr
c4.large	$0.191/hr	$0.105/hr	$0.296/hr
c4.xlarge	$0.23/hr	$0.209/hr	$0.439/hr
c4.2xlarge	$0.463/hr	$0.419/hr	$0.882/hr
c4.4xlarge	$0.614/hr	$0.838/hr	$1.452/hr
c4.8xlarge	$0.763/hr	$1.675/hr	$2.438/hr
t2.large	$0.131/hr	$0.104/hr	$0.235/hr
m4.large	$0.288/hr	$0.12/hr	$0.408/hr
m4.xlarge	$0.49/hr	$0.239/hr	$0.729/hr
m4.2xlarge	$0.557/hr	$0.479/hr	$1.036/hr
m4.4xlarge	$0.653/hr	$0.958/hr	$1.611/hr
m4.10xlarge	$1.09/hr	$2.394/hr	$3.484/hr

Ilustración 69: Costos de Wowza en Amazon en US East Virginia. Fuente: "AWS Marketplace: Wowza Streaming Engine 4: Pro Edition (HVM)", s/f.

Adobe Media Server necesita correr en servidores con mucho más RAM y de alta rendimiento. Por lo cual solo se muestra en servidores de alta gama. Wowza permite ser manejado en servidores mucho más pequeños y tiene una gama más grande para elegir. Los costos son similares. Debido a las pruebas que se hizo para este proyecto recomendamos usar por lo menos instancias medianas como t2.medium. Se pudo instalar y correr Red5 en instancias mucho más pequeñas pero en pruebas menores y en muchos casos se colgaron las aplicaciones.

Si hacemos un análisis de costo de Red5 vs Adobe Media Server en costos a largo plazo se puede ver una diferencia significativa de costos.

Tipo de Instancia	Software	EC2	Total/Hora	Costo EC2/Mes	Costo Total/ Mes	Diferencia
m1.large	0.23	0.175	0.405	126	291.6	165.6
m1.xlarge	0.97	0.35	1.32	252	950.4	698.4
m2.xlarge	0.23	0.245	0.475	176.4	342	165.6
m2.2xlarge	3.76	0.49	4.25	352.8	3060	2707.2
m2.4xlarge	3.76	0.98	4.74	705.6	3412.8	2707.2
m3.large	0.23	0.133	0.363	95.76	261.36	165.6
m3.xlarge	0.46	0.266	0.726	191.52	522.72	331.2

Tabla 2: Diferencia de costos de Adobe Media Server vs Red5. Elaboración propia..

Si se analiza la tabla superior podemos ver que la diferencia de costos viene a ser el costo de uso por el software. Un servidor con Red5 instalado en esas instancias podría llegar a costar 2700 dólares menos en un mes. En los casos menores costaría 165 dólares menos por mes. Esto es una diferencia significativa para empresas pequeñas.

Red5 vs Wowza:

Tipo de Instancia	Software	EC2	Total/Hora	Costo EC2/Mes	Costo Total/ Mes	Diferencia
t2.small	0.157	0.026	0.183	18.72	131.76	113.04
t2.medium	0.147	0.052	0.199	37.44	143.28	105.84
m3.medium	0.188	0.067	0.255	48.24	183.6	135.36
m3.large	0.276	0.133	0.409	95.76	294.48	198.72
m3.xlarge	0.488	0.266	0.754	191.52	542.88	351.36
m3.2xlarge	0.546	0.532	1.078	383.04	776.16	393.12
c3.large	0.181	0.105	0.286	75.6	205.92	130.32
c3.xlarge	0.201	0.21	0.411	151.2	295.92	144.72
c3.2xlarge	0.401	0.42	0.821	302.4	591.12	288.72
c3.4xlarge	0.531	0.84	1.371	604.8	987.12	382.32
c3.8xlarge	0.661	1.68	2.341	1209.6	1685.52	475.92
r3.large	0.29	0.166	0.456	119.52	328.32	208.8
r3.xlarge	0.459	0.333	0.792	239.76	570.24	330.48
r3.2xlarge	0.661	0.665	1.326	478.8	954.72	475.92
r3.4xlarge	0.79	1.33	2.12	957.6	1526.4	568.8
r3.8xlarge	1.09	2.66	3.75	1915.2	2700	784.8
c4.large	0.191	0.105	0.296	75.6	213.12	137.52
c4.xlarge	0.23	0.209	0.439	150.48	316.08	165.6
c4.2xlarge	0.463	0.419	0.882	301.68	635.04	333.36
c4.4xlarge	0.614	0.838	1.452	603.36	1045.44	442.08
c4.8xlarge	0.763	1.675	2.438	1206	1755.36	549.36
t2.large	0.131	0.104	0.235	74.88	169.2	94.32
m4.large	0.288	0.12	0.408	86.4	293.76	207.36
m4.xlarge	0.49	0.239	0.729	172.08	524.88	352.8
m4.2xlarge	0.557	0.479	1.036	344.88	745.92	401.04
m4.4xlarge	0.653	0.958	1.611	689.76	1159.92	470.16
m4.10xlarge	1.09	2.394	3.484	1723.68	2508.48	784.8

Tabla 3: Diferencia de costos de Wowza vs Red5. Elaboración propia.

Para servidores pequeños Wowza tiene unos gastos un poco menores a los de Adobe Media Server, pero sigue siendo cerca de los $105 de diferencia. Para una empresa pequeña esto llega a ser un gasto significante en ahorro o gasto.

Durante la investigación de este trabajo se encontraron algunas tendencias sobre la calidad de la señal y/o cantidad de conexiones manejadas de Red5 vs Wowza o Red5 vs Adobe Media Server en el Internet.

5.3.2 Análisis de las opiniones sobre Red5

Durante la investigación de este trabajo encontré algunas tendencias sobre la calidad de la señal y/o cantidad de conexiones manejadas de Red5 vs Wowza o Red5 vs Adobe Media Server en el Internet.

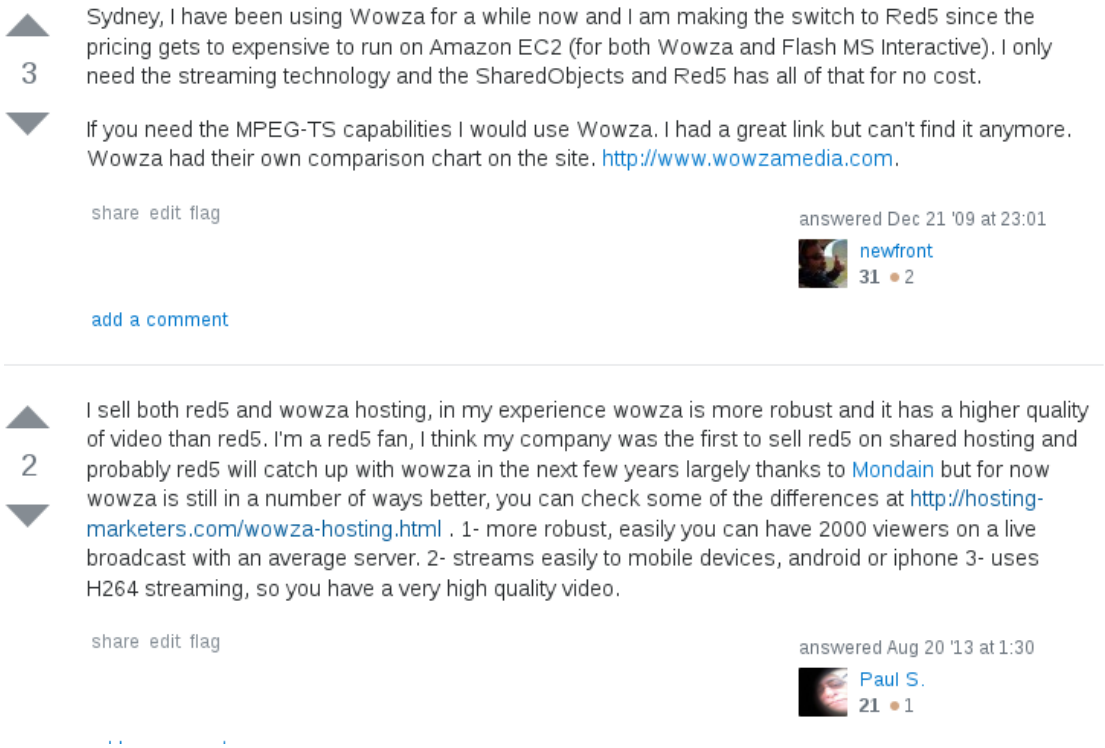

Ilustración 70: Nociones de Red5 en StackOverflow. Fuente: "Whats difference between FMS, Wowza and Red5, any comparison table would be helpfull - Stack Overflow", s/f.

La opinión general es que Red5 aún no está al nivel de producción de sus contrapartes comerciales, como se puede ver en los comentarios. Por otro lado, no se ha encontrado ninguna data que respalde esto directamente.

En las pruebas que se hicieron no pasaron más de 10 conexiones simultáneas por lo cual no se pudo manejar una prueba de estrés necesarias por falta de infraestructura. No se tiene el suficiente ancho de banda para manejar todas esas conexiones desde un solo punto de conexión, por lo cualquier conexión de Internet local colapsa antes de que empiecen a fallar el servidor.

No se pudo notar ningún cambio significativo entre Red5, Wowza o Adobe Media Server. Tal vez en producciones a más grande escala se pueda notar las diferencias.

La mayor diferencia que se puede notar es que los servidores de pago si soportan mucho más servicios y protocolos que Red5. La documentación de Red5 es escasa y en muchos casos desactualizada.

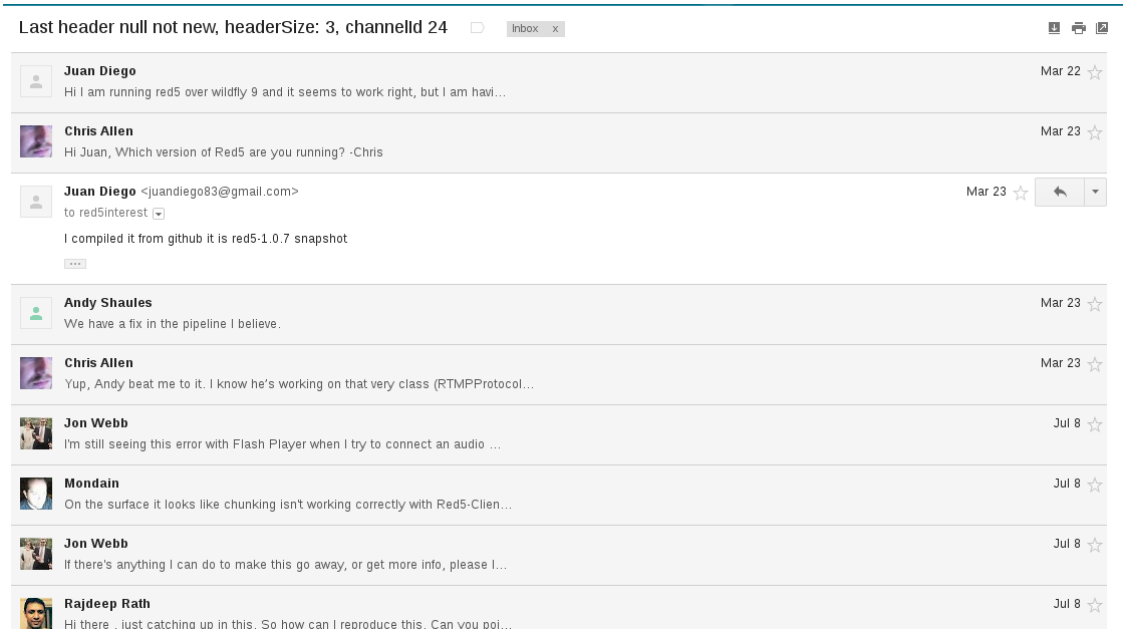

Ilustración 71: Respuestas y manejo de la comunidad de Red5 al reportar un error.

Cuando se tuvo problemas con Red5 la comunidad siempre respondió. Aunque en algunos casos puede tomar unos días. En el caso de la gráfica superior sucedió cuando se reportó un problema con una instalación y resultó ser un bug que fue corregido casi inmediatamente. Es interesante que en algunos casos hasta respondieron los desarrolladores y hasta el creador del proyecto Paul Mondain.

Con esto último se da por concluido el proyecto ya que se ha demostrado que es posible el proyecto.

Capítulo 6: Conclusiones y Recomendaciones

6.1 Conclusiones

Este trabajo se enfocó en la solución por parte del servidor para obtener una solución de alta disponibilidad porque es la propuesta a la que se quería llegar. Con ello se cumplió el objetivo general de estudiar el balanceo de carga de un sistema de software libre para streaming de alta disponibilidad utilizando cluster de Jboss con Red5 en la nube con Amazon Web Services (AWS).

Se cumplieron los objetivos específicos mediante el desarrollo de los diversos capítulos. La implementación de la propuesta es el aporte principal de este trabajo por las adaptaciones que fue necesario realizar y que dejan el camino abierto para nuevas investigaciones.

En este sentido, para siguientes estudios se podría considerar el lado del cliente que seguramente enriquecerá las soluciones al realizar sus propios proyectos, por el tema tratado queda abierto para que se continúe en estudios posteriores.

Las conclusiones a las que se ha llegado a nivel técnico son:

- Se ha comprobado la hipótesis: es posible crear un streaming de alta disponibilidad con alta velocidad con clusters de JBoss con Red5 que puede ser aplicado par video y audio esencialmente aunque se podría aprovechar para programas más complejos que utilicen tiempo real.
- Red5 con Wildfly/JBoss provee una solución de bajo costo con software libre y/o de código abierto que es totalmente aplicable a un cluster de alta disponibilidad para streaming, motivo de esta tesis.
- Al ser Red5 software libre permite ser modificado, lo cual es una ventaja para una empresa pequeña siempre y cuando se tenga el conocimiento necesario. De esta forma permite aprender y entender cómo se manejan.
- Las comunidades activas de software libre/código abierto permiten a las empresas pequeñas manejar productos de calidad que compiten contra productos pagos de alto costo y alto rendimiento.
- En el caso de esa investigación estas comunidades fueron esenciales en razón de la poca documentación disponible para Red5.
- Wildfly/JBoss es un servidor de aplicaciones sumamente amplio que permite crear soluciones de alta disponibilidad en varias plataformas y servicios. Es perfecto para manejar aplicaciones en la nube. No solo tiene una comunidad a nivel mundial para su desarrollo sino que tiene el respaldo de empresas como Red Hat y por tanto fue indispensable para el desarrollo del cluster.

- Una implementación total de Red5 con Wildfly/JBoss con Infinispan provee una gran ventaja para el manejo de aplicaciones de streaming. En el caso de este estudio fue necesario realizar adaptaciones del Red5 para que pueda utilizar Infinistan lo que significó un aumento de complejidad al momento de realizar las aplicaciones.

6.2 Recomendaciones

- Red5 es difícil de manejar, requiere de una experiencia amplia en Java, conocimientos de streaming. No es recomendable para un proyecto que tenga un tiempo limitado sin poder hacer muchas pruebas antes y sin experiencia previa.
- Para mejorar estas aplicaciones es necesario utilizar los “Listeners” de Infinispan y Jgroups con mayor detenimiento lo cual permite manejar y corregir problemas que se dan al agregar o quitar nodos.
- Los requerimientos de streaming altos en cuanto a transferencia de datos, procesamiento y RAM, lo cual se debe tener en cuenta los costos de cada uno de estos rubros para hacer proyecciones eficientes de gastos.
- Poder hacer pruebas de estrés con RTMP sería muy beneficioso para lo cual se requiere una infraestructura mayor.
- Utilizar Red5 con una CDN o red de distribución de contenido con Red5 sería de mucha ayuda, aunque teniendo varios servidores de Red5 en distintas zonas permitiría crear un CDN.
- Sería mucho mejor si Red5 por defecto tendría contenido Infinispan para poder no tener que hacer modificaciones extra.
- El uso del inglés es necesario para Red5. Tomaría mucho más tiempo esperar a que se traduzca la documentación necesaria.
- Para bajar costos en las pruebas se recomienda usar interfaces virtuales.
- Para proyectos que tengan limitaciones de tiempo y requieran un servicio de soporte garantizado el software de pago como Adobe Media Server y Wowza es más conveniente de utilizarlo ya que estos dispositivos de pago requieren de una menor configuración.

Fuentes Citadas

AWS Marketplace: Adobe Media Server 5 Extended. (s/f). Recuperado el 28 de octubre de 2016, a partir de https://aws.amazon.com/marketplace/pp/B00IGJMIOI

AWS Marketplace: Wowza Streaming Engine 4: Pro Edition (HVM). (s/f). Recuperado el 28 de octubre de 2016, a partir de https://aws.amazon.com/marketplace/pp/B012BW3WB8?qid=1477695156016&sr=0-1&ref_=srh_res_product_title

Butler, B. (2015, mayo 21). Gartner: Amazon's cloud is 10x bigger than its next 14 competitors, combined. Recuperado el 18 de diciembre de 2016, a partir de http://www.networkworld.com/article/2925186/cloud-computing/gartner-amazon-s-cloud-is-10x-bigger-than-its-next-14-competitors-combined.html

Cluster (informática). (2016, mayo 12). En *Wikipedia, la enciclopedia libre*. Recuperado a partir de https://es.wikipedia.org/w/index.php?title=Cl%C3%BAster_(inform%C3%A1tica)&oldid=91008130

Configure Sticky Sessions for Your Load Balancer - Elastic Load Balancing. (s/f). Recuperado el 24 de junio de 2016, a partir de http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-sticky-sessions.html

EAR (file format). (2016, julio 5). En *Wikipedia, the free encyclopedia*. Recuperado a partir de https://en.wikipedia.org/w/index.php?title=EAR_(file_format)&oldid=728456605

Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS. (s/f). Recuperado el 18 de diciembre de 2016, a partir de //aws.amazon.com/ec2/

Elastic JBoss AS 7 clustering in AWS using EC2, S3, ELB and Chef - Amazon Web Services Germany. (2013, octubre 10). Recuperado el 9 de julio de 2016, a partir de http://aws.typepad.com/awsaktuell/2013/10/elastic-jboss-as-7-clustering-in-aws-using-ec2-s3-elb-and-chef.html

How To Install WildFly as a Service on Linux. (s/f). Recuperado el 14 de julio de 2016, a partir de http://developer-should-know.com/post/112230363742/how-to-install-wildfly-as-a-service-on-linux

Inc, T. (s/f). Periscope. Recuperado el 18 de diciembre de 2016, a partir de https://www.periscope.tv

Infinispan Homepage - Infinispan. (s/f). Recuperado el 9 de julio de 2016, a partir de http://infinispan.org/

JGroups - The JGroups Project. (s/f). Recuperado el 8 de julio de 2016, a partir de http://jgroups.org/

Maven – Welcome to Apache Maven. (s/f). Recuperado el 26 de julio de 2016, a partir de https://maven.apache.org/

Mozaffari, B. (2013, noviembre). JBoss EAP 6 Clustering, JBoss Enterprise Application Platform 6.1: High Availability, configuration and best practices. Red Hat. Recuperado a partir de https://access.redhat.com/sites/default/files/attachments/eap6_clustering_2_0.pdf

Red5/red5-examples. (s/f). Recuperado el 15 de julio de 2016, a partir de https://github.com/Red5/red5-examples

Red5/red5-io. (s/f). Recuperado el 27 de julio de 2016, a partir de https://github.com/Red5/red5-io

Red5/red5-parent. (s/f). Recuperado el 15 de julio de 2016, a partir de https://github.com/Red5/red5-parent

Red5/red5-rtsp-restreamer. (s/f). Recuperado el 19 de octubre de 2016, a partir de https://github.com/Red5/red5-rtsp-restreamer

Red5/red5-server. (s/f). Recuperado el 15 de julio de 2016, a partir de https://github.com/Red5/red5-server

Red5/red5-server-common. (s/f). Recuperado el 15 de julio de 2016, a partir de https://github.com/Red5/red5-server-common

Red5/red5-service. (s/f). Recuperado el 15 de julio de 2016, a partir de https://github.com/Red5/red5-service

Streaming. (2016, mayo 26). En *Wikipedia, la enciclopedia libre*. Recuperado a partir de https://es.wikipedia.org/w/index.php?title=Streaming&oldid=91301256

Twitch Payment Information. (2016, junio 14). Recuperado el 14 de junio de 2016, a partir de https://www.twitch.tv/products/turbo/ticket/new

Whats difference between FMS, Wowza and Red5, any comparison table would be helpfull - Stack Overflow. (s/f). Recuperado el 31 de octubre de 2016, a partir de http://stackoverflow.com/questions/1199890/whats-difference-between-fms-wowza-and-red5-any-comparison-table-would-be-help

WildFly. (2016, mayo 15). En *Wikipedia, la enciclopedia libre*. Recuperado a partir de https://es.wikipedia.org/w/index.php?title=WildFly&oldid=91071821

Glosario

Cluster: agrupamiento de servidores con similares configuraciones y aplicaciones que sirve generalmente para compartir y distribuir una carga.

DDoS: Distributed denial of service o ataque de negación de servcio distribuido

MMOG: Massive multiplayer online games en inglés o juegos masivos multijugador en línea.

Microblogging: servicio de publicación de mensajes cortos en el Internet.

RTMP: Es un protocolo de alto rendimiento diseñado para la transmisión con audio, video y datos. Es una tecnología con especificaciones abierta.

Streaming: transmición de audio, video o datos.